14 天学会安卓开发

作者:神秘的 N (英文名 corder_raine)
联系方式:369428455
交流群:284552167

前言:

本人也是菜鸟,老鸟看了此文有哪里不好之处敬请指点,本书是根据<<Android 应用开发揭秘>>攒写的,如何把一本书读薄,是一件值得思考的问题.本书针对有 JAVA 基础的孩纸们,基本 JAVA 基础都没的,赶紧去找 21 天学会 JAVA 回去从头开始学,或者找一本从零开始学 android 开发看也行.废话不多说,本人也是刚开始写,有哪里我写不清楚的请多多指点,原型是写出来了,后续慢慢更新,先把第一二天上传上来,看看大家有什么反应.

第一天 Android 架构与环境搭建

1.1 android 基础

1.1.1 Android 是什么?

- 是一个针对移动设备的操作系统和软件平台
- 基于 Linux 内核
- 由 Google 和开放手机联盟 OHA 开发的
- 容许使用 Java 语言来开发和管理代码
- Android 开放源代码,Android 遵从 Apache Software License (ASL)2.0 版本的协议
- Android 于 2007 年 11 月 5 日开放手机联盟成立时发布

注:开放手机联盟 (OHA)

1.1.2 Android 的特点!

- 应用框架可以重复使用，其组件也可以更换。
- **Dalvik** 虚拟机针对移动设备进行了优化。
- 优化的图形能力支持 **2D**、**3D** 图形**(OpenGL ES 1.0)**。
- 集成了基于开源 **WebKit** 引擎的浏览器。
- **SQLite** 作为结构化数据存储。
- 多媒体支持多种音频、视频格式。
- **GSM Telephony (hardware dependent)**
- 支持蓝牙 **Bluetooth**，**3G** 和 **WiFi**
- 支持照相机、**GPS**、指南针和加速度仪等传感器硬件。
- 丰富的开发环境。包括模拟机、调试工具、内存运行检测，以及为 **Eclipse IDE** 所写的插件。

1.1.3 Android 优缺点!

优点

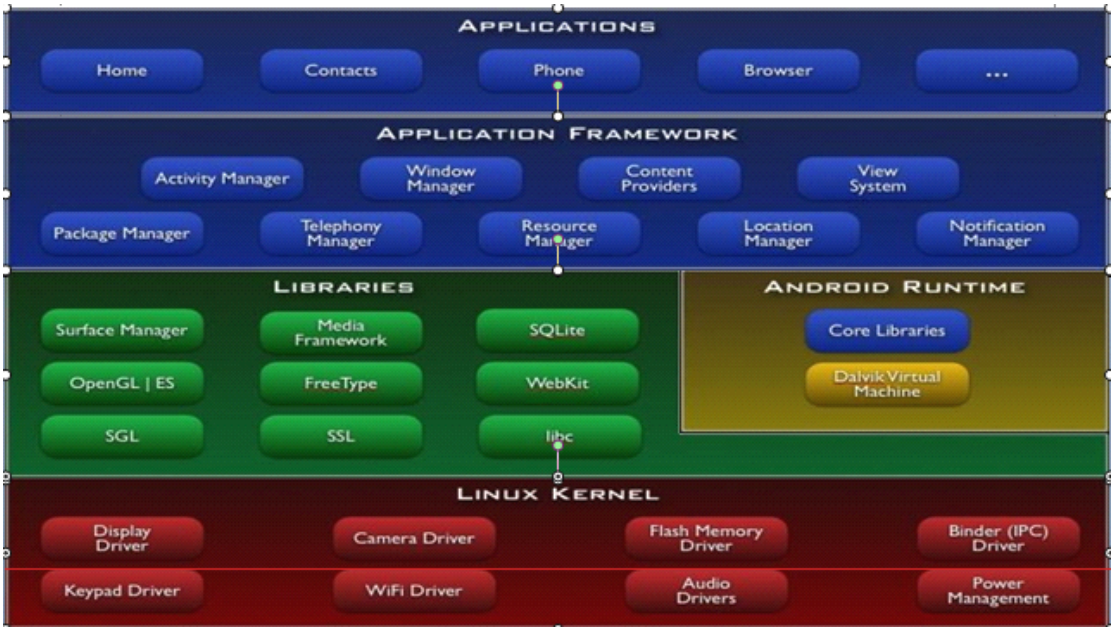
- 源代码完全开放。
- 采用了对有限内存、电池和 **CPU** 优化过的虚拟机 **Dalvik**，**Android** 的运行速度比想象的要快很多。
- 运营商（中国移动等）的大力支持，产业链条的热捧。

- 良好的盈利模式（**3/7** 开），产业链条的各方：运营商、制造商、独立软件生产商都可以获得不错的利益。将移动终端的评价标准从硬件向软件转变，极大的激发了软件开发者的热情。
- **Android** 的源代码遵循 **Apache V2** 软件许可，而不是通常的 **GPL v2** 许可。有利于商业开发。
- 具有强大的 **Linux** 社区的支持。

缺点

- **Google** 提供了一套 **Java** 核心包**(J2SE 5,J2SE 6)**的有限子集，尚不承诺遵守 **Java** 任何 **Java** 规范,可能会造成 **Java** 阵营的进一步分裂。
- 现有应用完善度不太够，需要的开发工作量较大。
- 模拟器调试手段不十分丰富，速度慢。

1.1.4 Android 架构



应用层

应用框架层
(开发调用)

运行机制
(DVM-JVM)

类库层

底层
(基于 linux 内核)

Linux Kernel& Android 内核

- **Android** 基于 **Linux** 内核，但不是 **Linux**
- 内核提供系统核心服务，如进程、内存、电源管理，网络连接，驱动与安全等。
- 并不包括全部的 **Linux**。
- **Linux** 内核位于硬件和软件堆之间的抽象层
  - 核心服务：安全机制、内存管理、进程管理、网络、硬件驱动。
  - 内核扮演的是硬件层和系统其它层次之间的一个抽象层的概念。
  - 操作系统的初始化和编程接口和标准的 **Linux** 系统是有所不同的。

Libraries

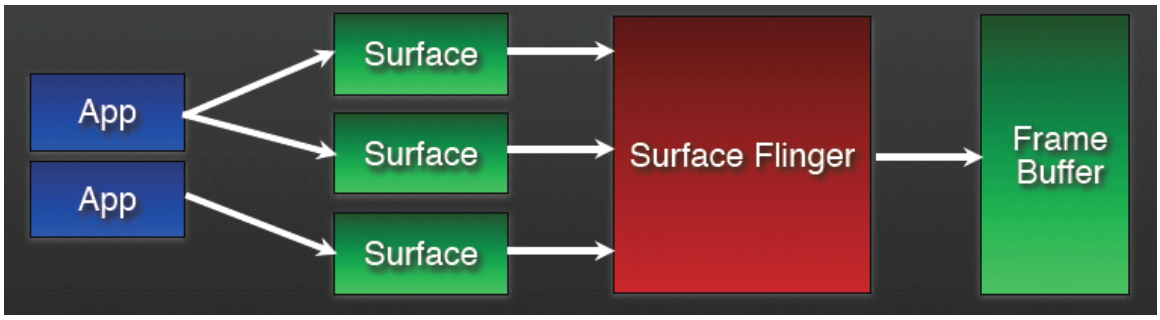
- **C/C++**库：被各种 **Android** 组件使用通过应用程序框架开发者可以使用其功能包括：
- 媒体库：**MPEG4 H.264 MP3 JPG PNG .....**
- **WebKit/LibWebCore**：**Web** 浏览引擎
- **SQLite** 关系数据库引擎
- **2D**，**3D** 图形库、引擎

Function Libraries

- **WebKit**
  - 基于开源 **WebKit** 的浏览器
  - 支持 **CSS**、**Javascript**、**DOM**、**Ajax**
- 多媒体框架
  - 基于 **PacketVideo OpenCORE** 平台
  - 支持标准音频、视频
- **SQLite**
  - 轻型数据库，支持多种平台

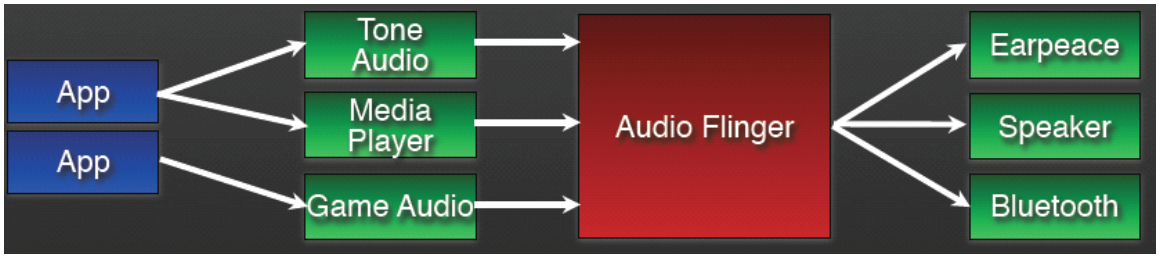
Native Servers-Surface

- 为多种应用提供 2D、3D 表面设计



Native Servers-Audio

- 音频处理



Hardware Abstraction Libs

- 硬件虚拟层
- **User space C/C++ library layer**
- 硬件接口驱动
- 使 **Android** 平台逻辑与硬件接口分离

DVM vs JVM

- **DVM**
  - **Google**
  - **Dalvik excutable**

  JVM
  - **Sun**
  - **Java bytecode**

Applications Framework

- **Activity manager**
  - 管理运行应用程序

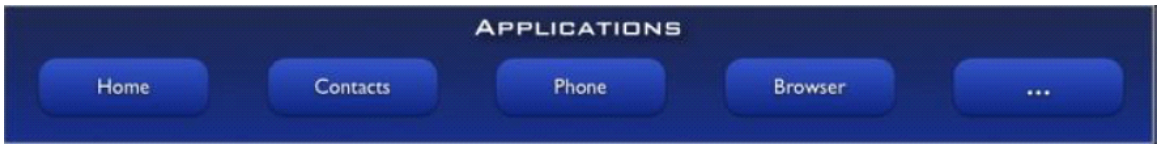
- **Content Provider**
  - 在各应用之间共享数据
- **Resource Manager**
  - 管理非代码资源
- **Notification Manager**
  - 显示用户提示和状态栏
- **Views System**
  - 可扩展显示，可构建 **UI**

应用和框架

- 核心应用，例如联系人，电子邮件，电话，浏览器，日历，地图， **...**
- 充分访问所有核心应用框架 **API**
- 简化组件的重用
- 用 **Java** 编写应用程序

Applications

- **JAVA** 编写的应用程序



1.2 搭建 android 环境

注:如果想从头开始学怎么搭建 android 开发环境的请从 1.2.1 开始
如果想直接学开发了,请跳过此步骤,到 1.2.6 节直接下载谷歌提供的开发工具进行开发.

1.2.1 安装 Java JDK 并配置 java 环境变量.

下载 JDK,(官方推荐使用 1.6 版本)
http://www.oracle.com/technetwork/java/javase/downloads/index.html(官网下载地址)
http://download.oracle.com/otn-pub/java/jdk/6u37-b06/jdk-6u37-windows-i586.exe
(x86 直接下载地址)
http://download.oracle.com/otn-pub/java/jdk/6u37-b06/jdk-6u37-windows-x64.exe
(x64 直接下载地址)

安装 JDK 略过

打开环境变量窗口方法：右键【我的电脑】--单击【属性】--单击【高级】--单击【环境变量】。
在上方的用户变量中依次新建如下变量，并分别填入如下路径：

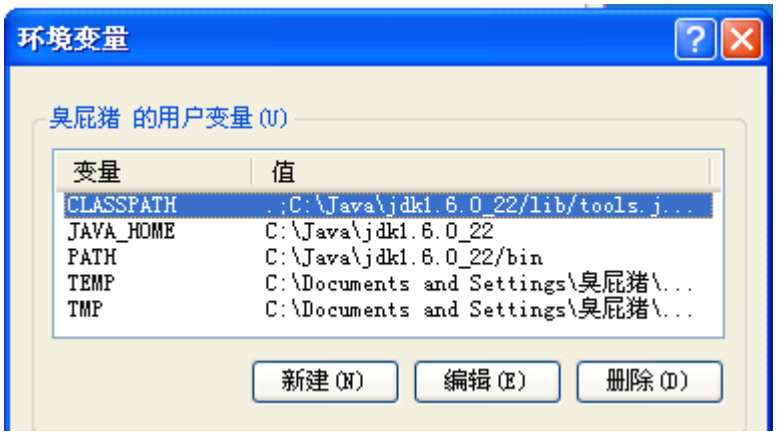
→变量名：JAVA_HOME
变量值：C:\Java\jdk1.6.0_22(此路径为 JAVA 安装路径)

→变量名：PATH

变量值：%JAVA_HOME%/bin

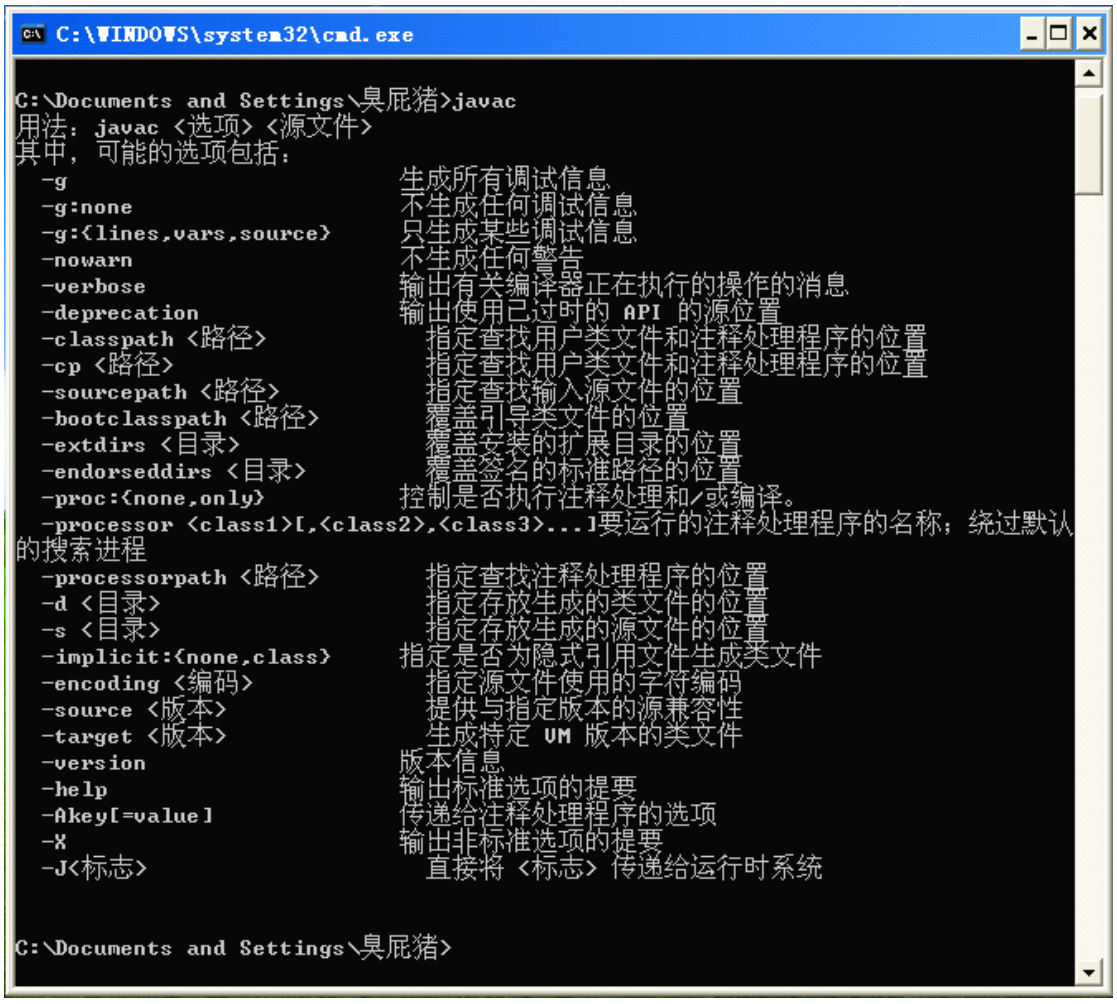
→变量名：CLASSPATH

变量值：.;%JAVA_HOME%/lib/tools.jar;%JAVA_HOME%/lib/dt.jar



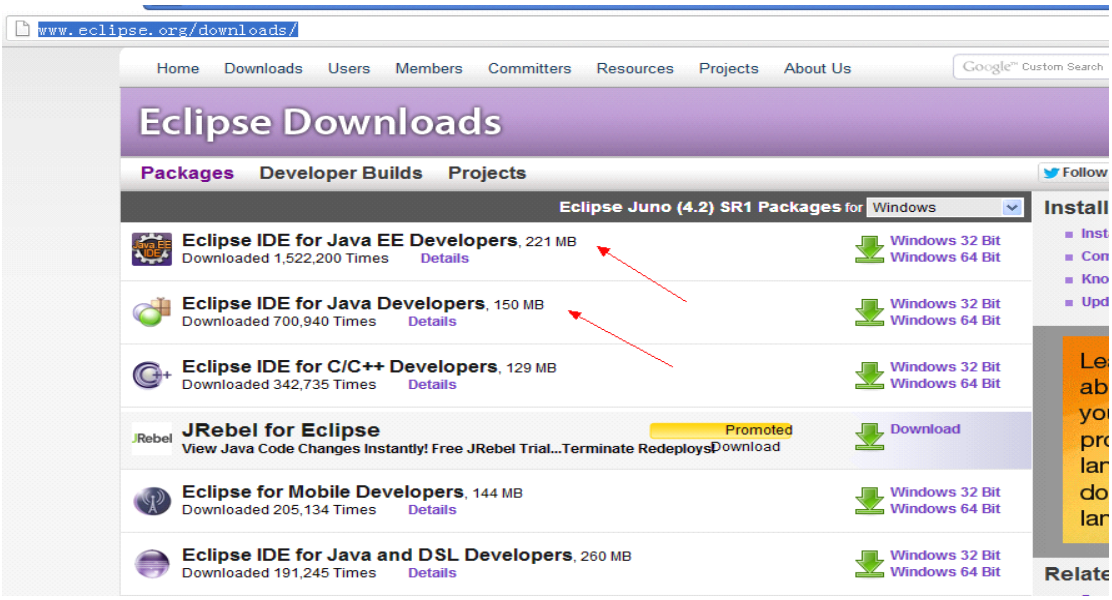
测试环境变量配置是否成功

【开始】--【运行】--输入【cmd】--输入【javac】--按【回车键】若看到以下信息，则代表配置成功。



1.2.2 下载 eclipse 安装 ADT 插件

Eclipse 下载地址:http://www.eclipse.org/downloads/(java 和 J2EE 都行)
http://build.eclipse.org/technology/phoenix/torrents/juno/eclipse-jee-juno-SR1-win32.zip.torrent
(x86 种子地址,如果下载不了请到官方下载)
http://build.eclipse.org/technology/phoenix/torrents/juno/eclipse-jee-juno-SR1-win32.zip.torrent
X64 的



安装 ADT

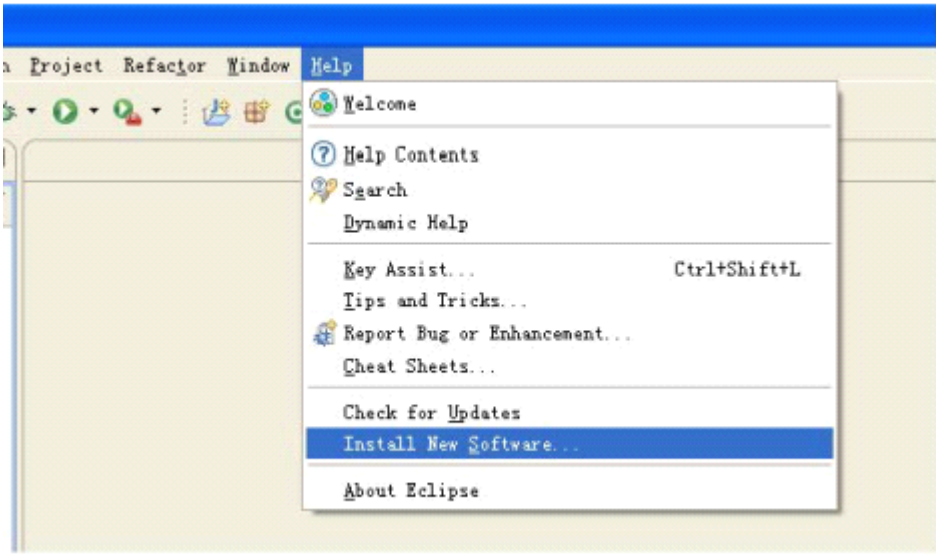
注意:

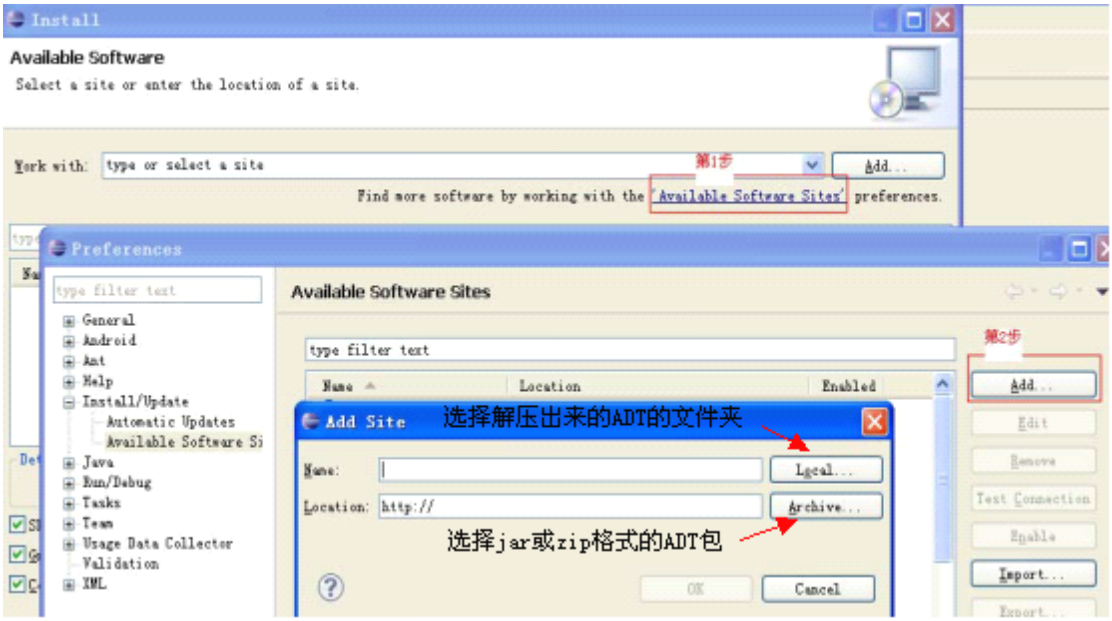
安装 ADT 的方法有两种

一 在线安装(输入 Name 和 Location 即可在线安装,不推荐使用,因为在线更新灰常慢)

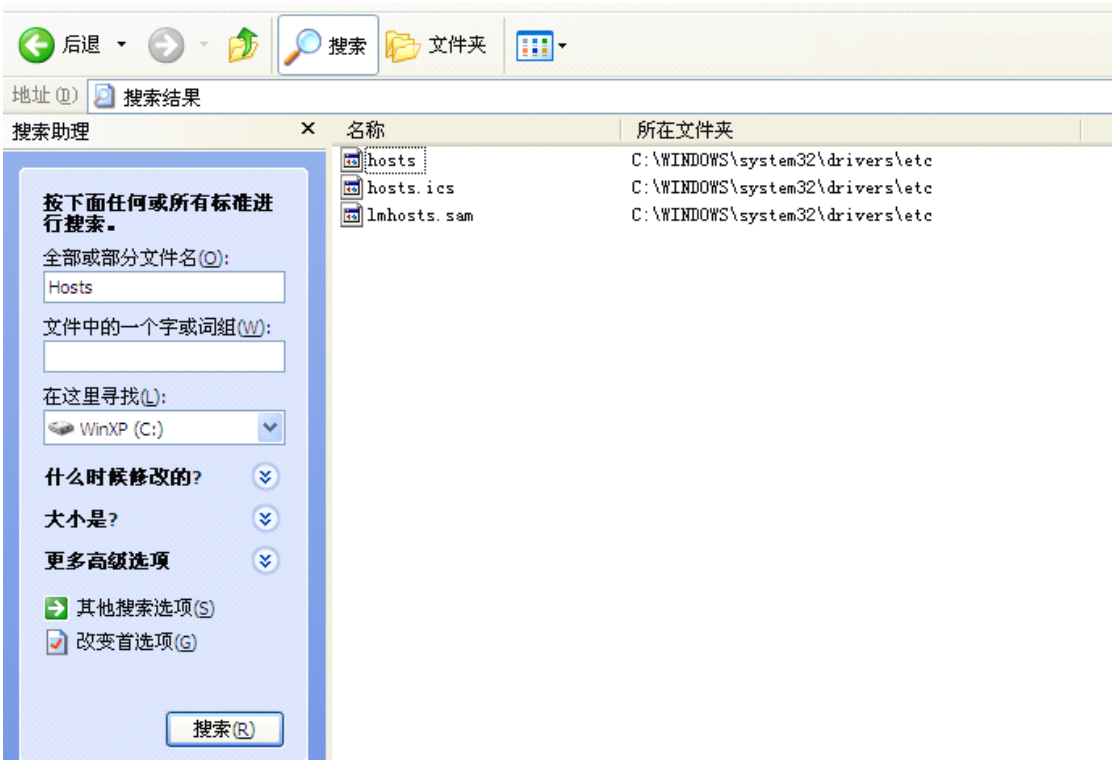
Name:Andriod Plugin

Location： https://dl-ssl.google.com/android/eclipse/

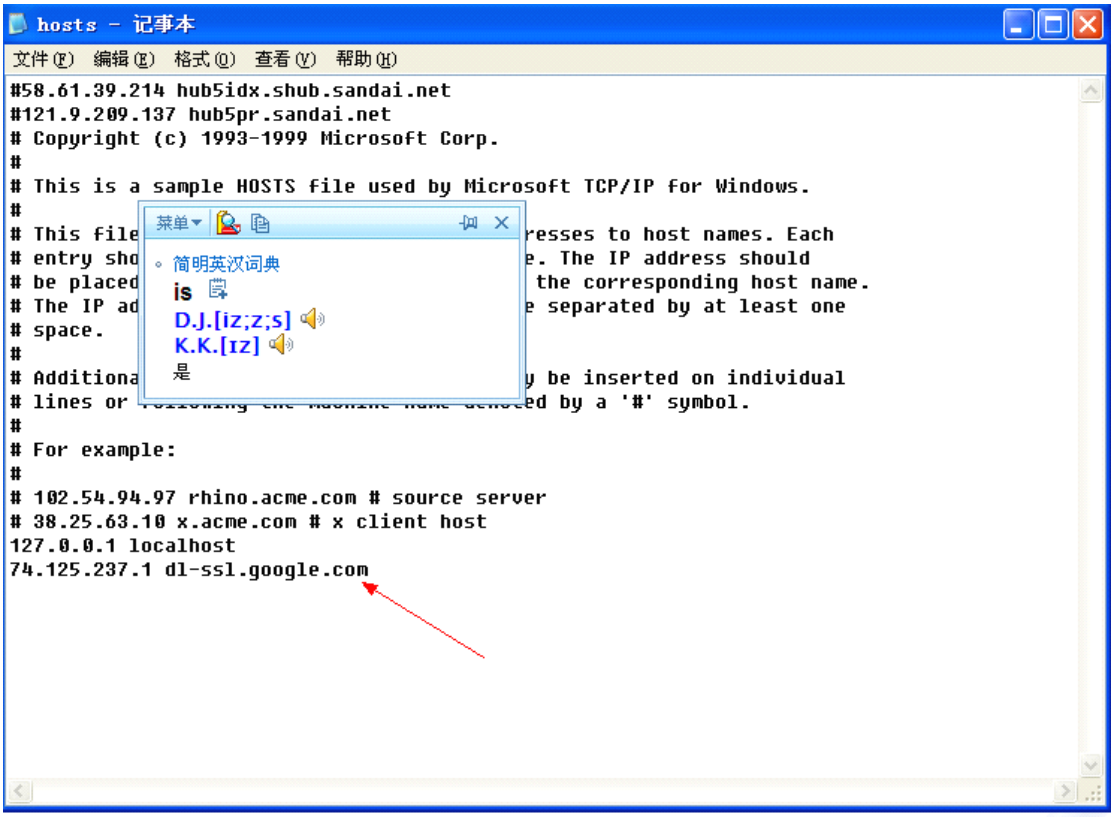




执意想要在线更新的朋友,我们只需要配置一下 hosts 文件(翻墙不用我说吧?)
Hosts 文件只需要在 C 盘搜索一下就出来了



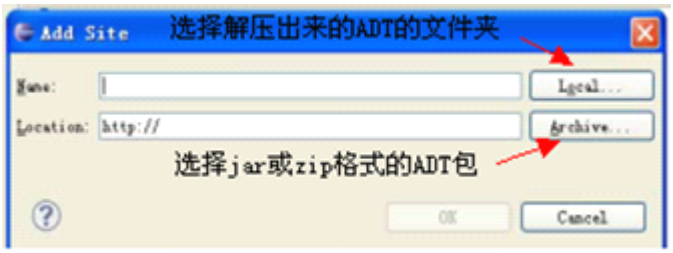
在文件的末尾添加下面一句：74.125.237.1 dl-ssl.google.com



(记得右键吧 hosts 文件只读的勾去了)

二　离线安装(离线安装要断网,不然它会自动联网安装的)

离线版 ADT 下载　http://developer.android.com/sdk/index.html



离线更新有两种方法

1 点 local..选择解压出来的 ADT 文件夹(好处是解包已在外部完成了,可以直接更新,不用等待,推荐使用)

2.点 archive 选择打包的 ADT(好处是不用在外部解包,但是不可直接更新,要等 eclipse 解包完 ADT.ZIP 才能进行安装,不推荐使用)

如果推荐使用的方法不行,哪就用其他方法吧!

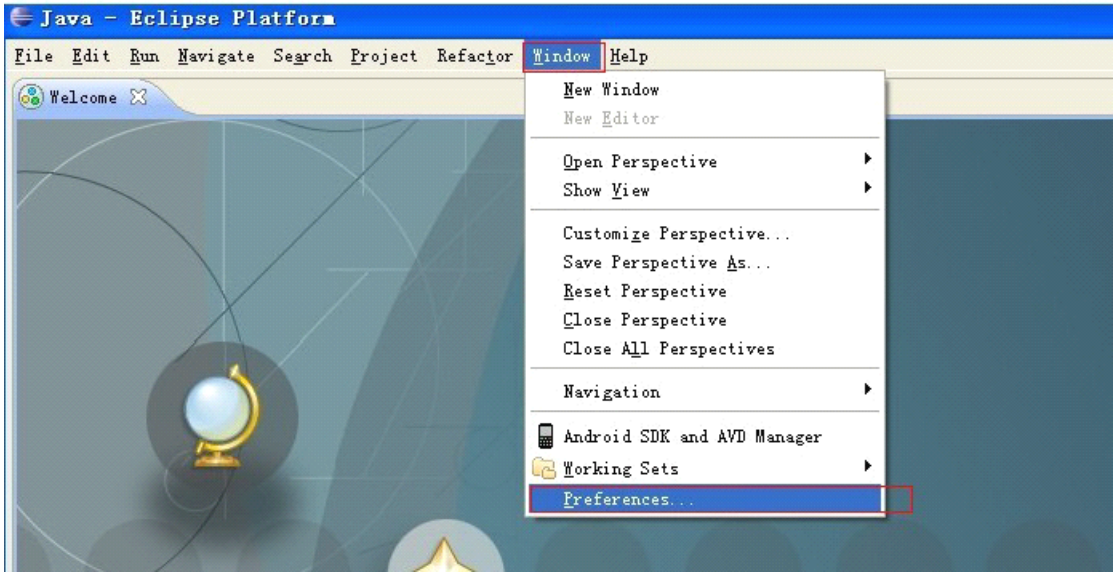
1.2.3 安装 android sdk 并更新版本

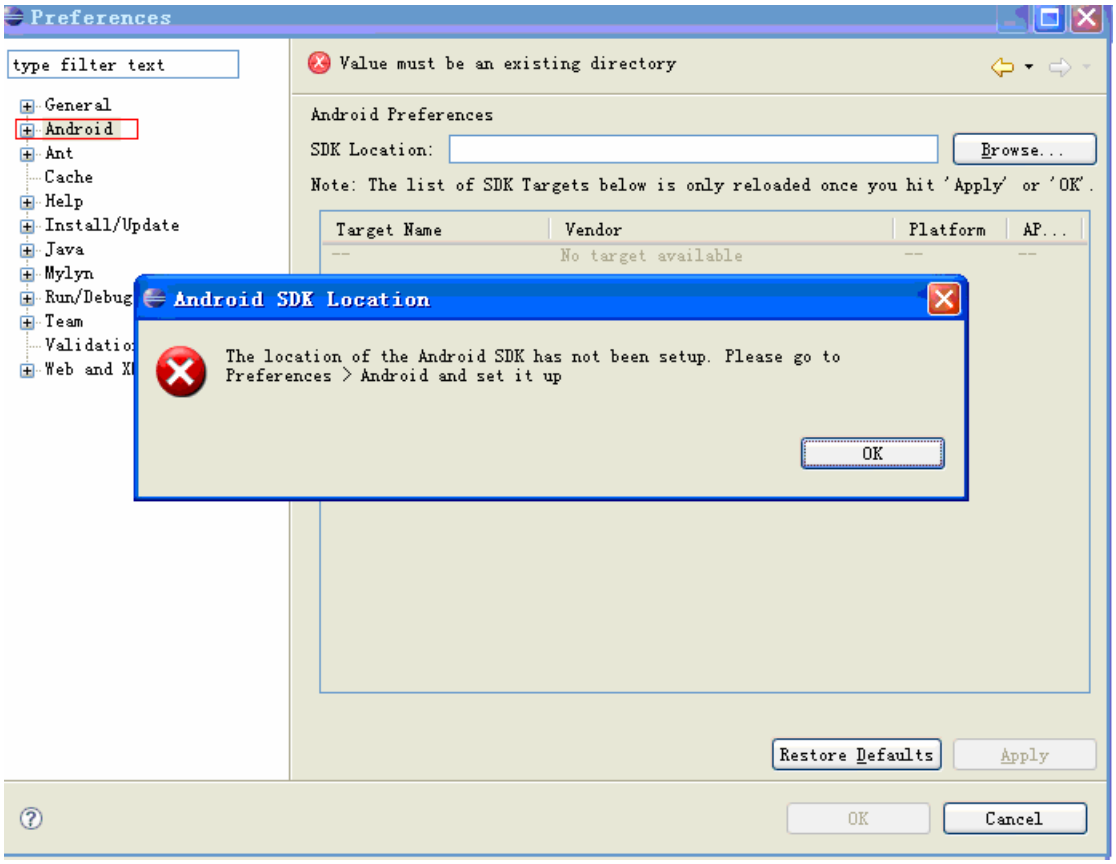
安装完 ADT 之后它会提示你重启 eclipse,重启完后会提示选择在线更新 SDK 还是选现有的 SDK(这里只说明选现有的 SDK)

注意:记得选第二个,默认的是让你选路径下载 SDK 的

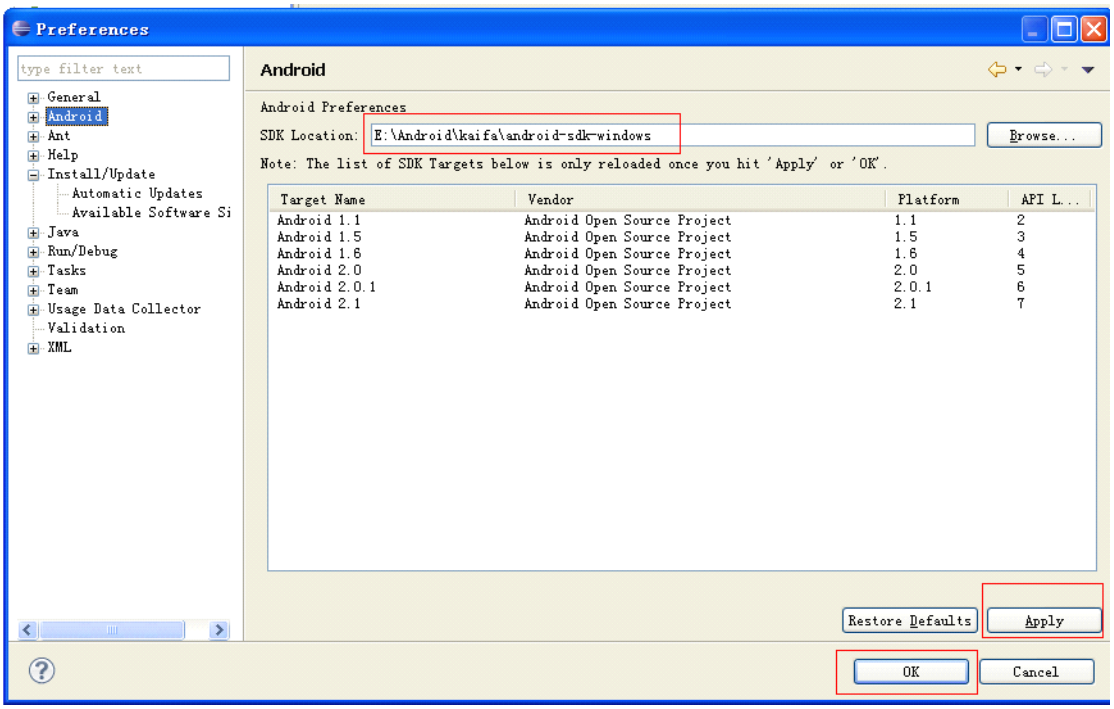
下载 SDK: http://developer.android.com/sdk/index.html

离线安装 Sdk

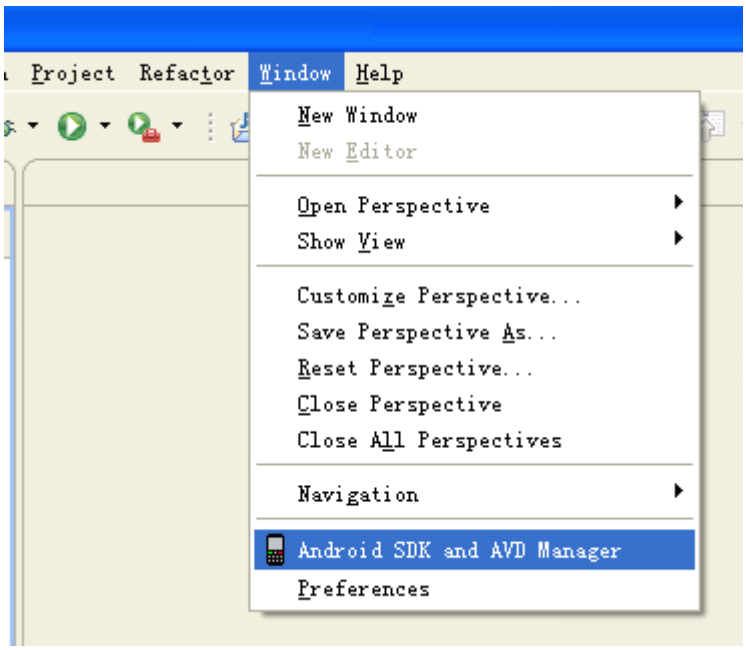


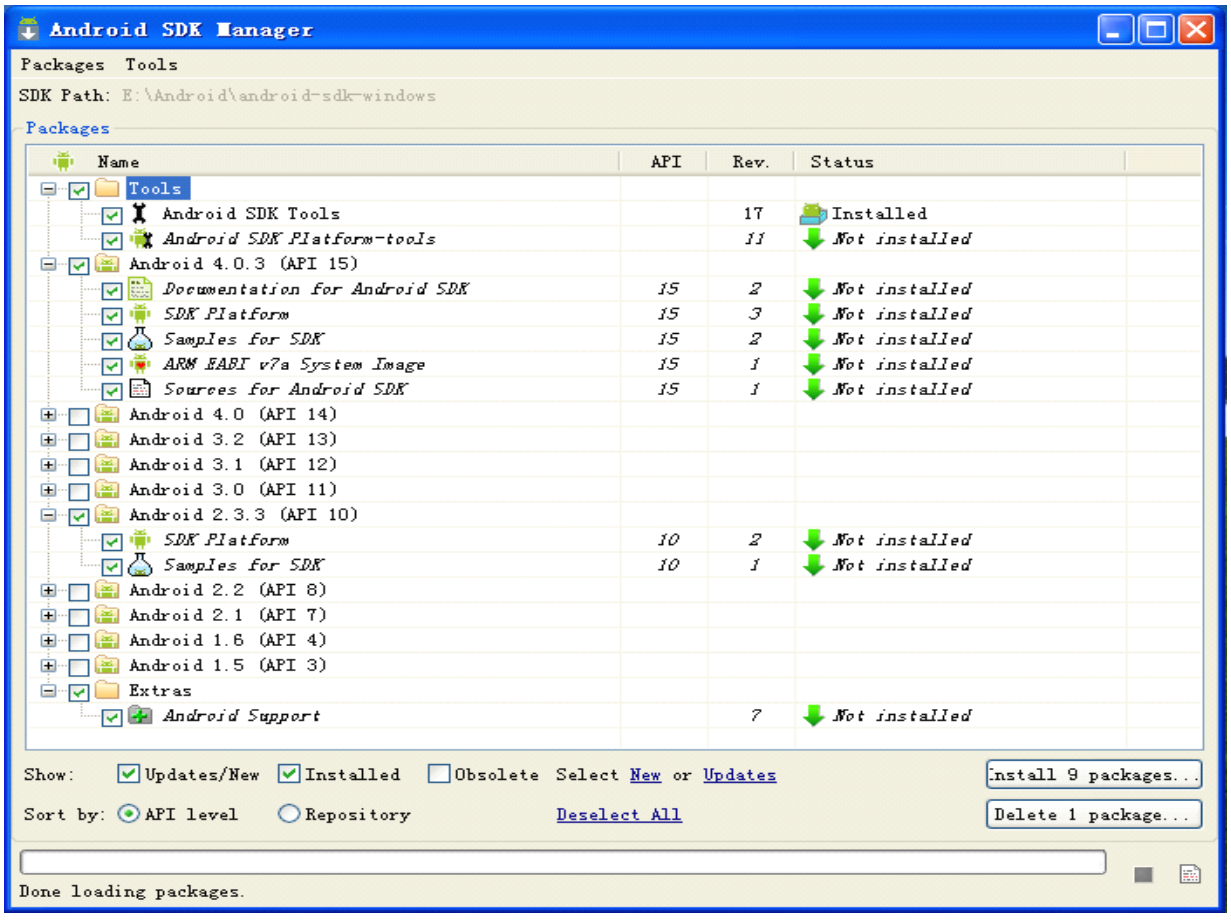


找到 SDK 路径应用一下就可以了



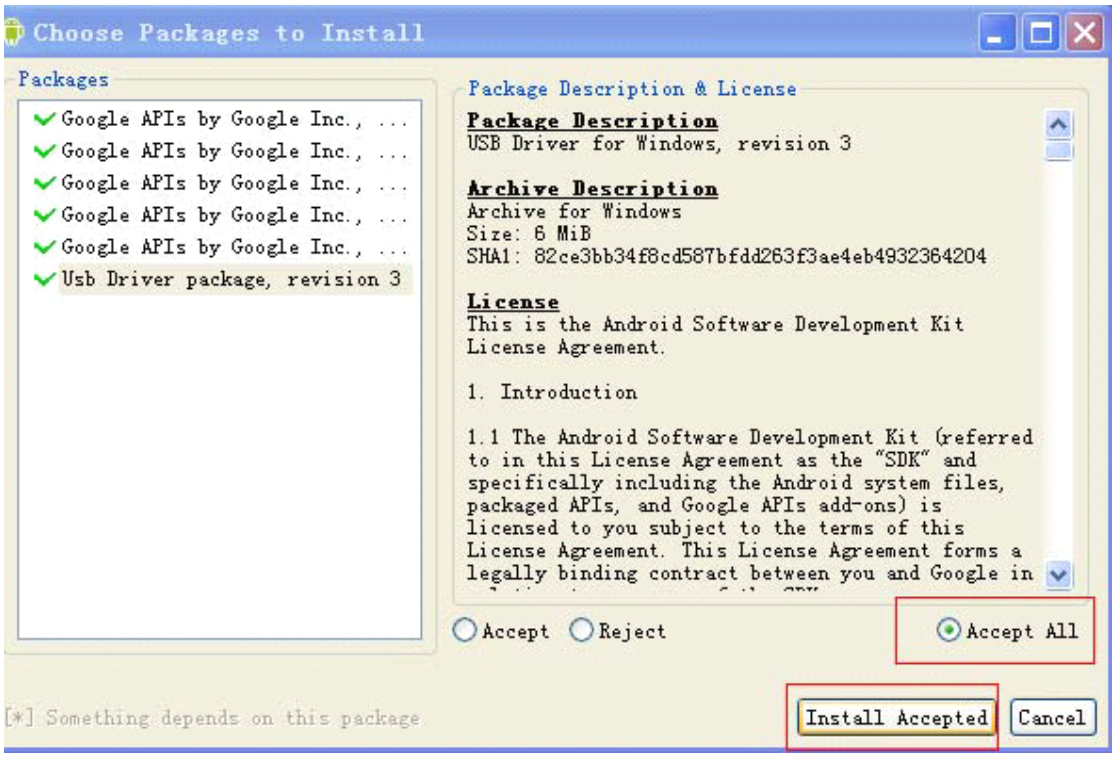
管理自己的 SDK,升级或者更新





在自己需要的版本上打勾然后按 install　(一般开发都用 2.1,谷歌会提示你 95%的开发者都在用 2.1 之类的话的)

点击 install 后如下图,选择所有 再点 install



更新完后就大功告成了

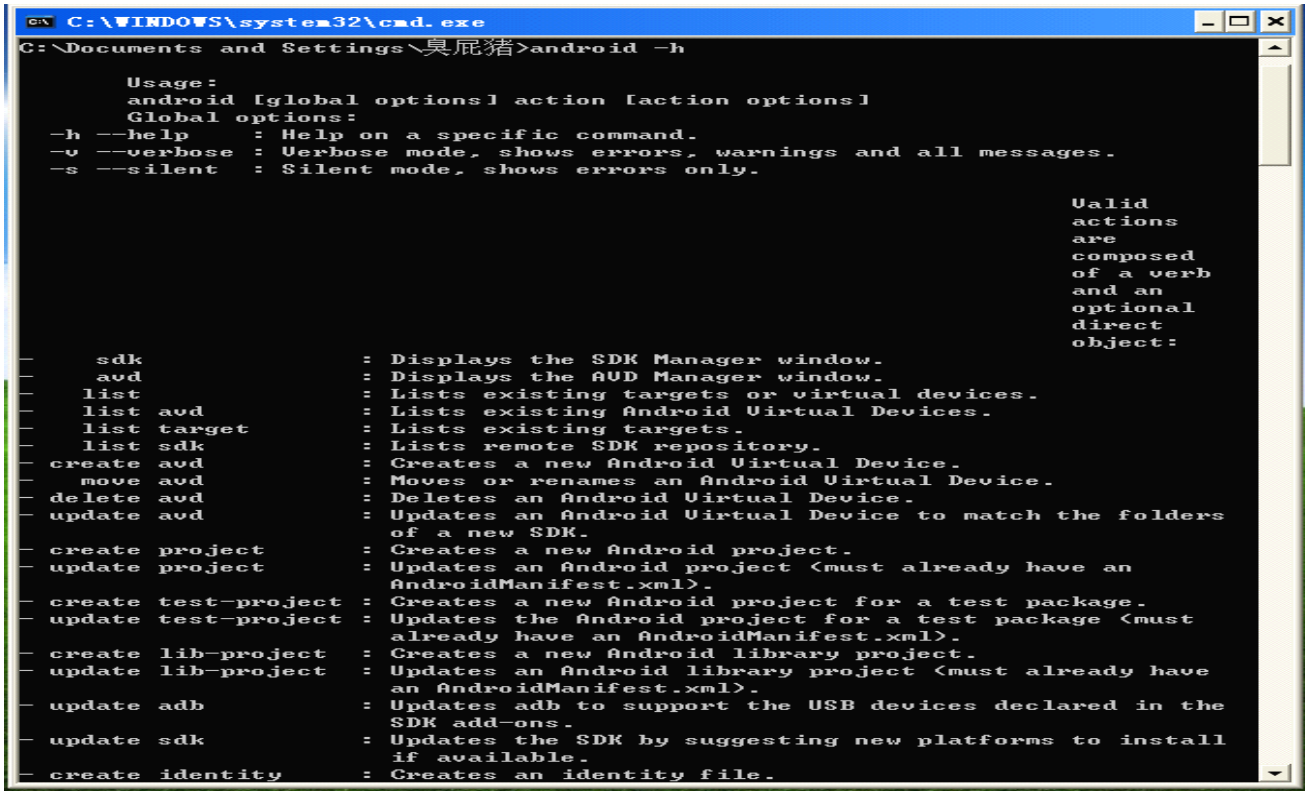
1.2.4 配置 android 环境变量.

在上方的用户变量中找到之前创建的【PATH】变量，双击它，然后在【变量值】的最后面添加上内容

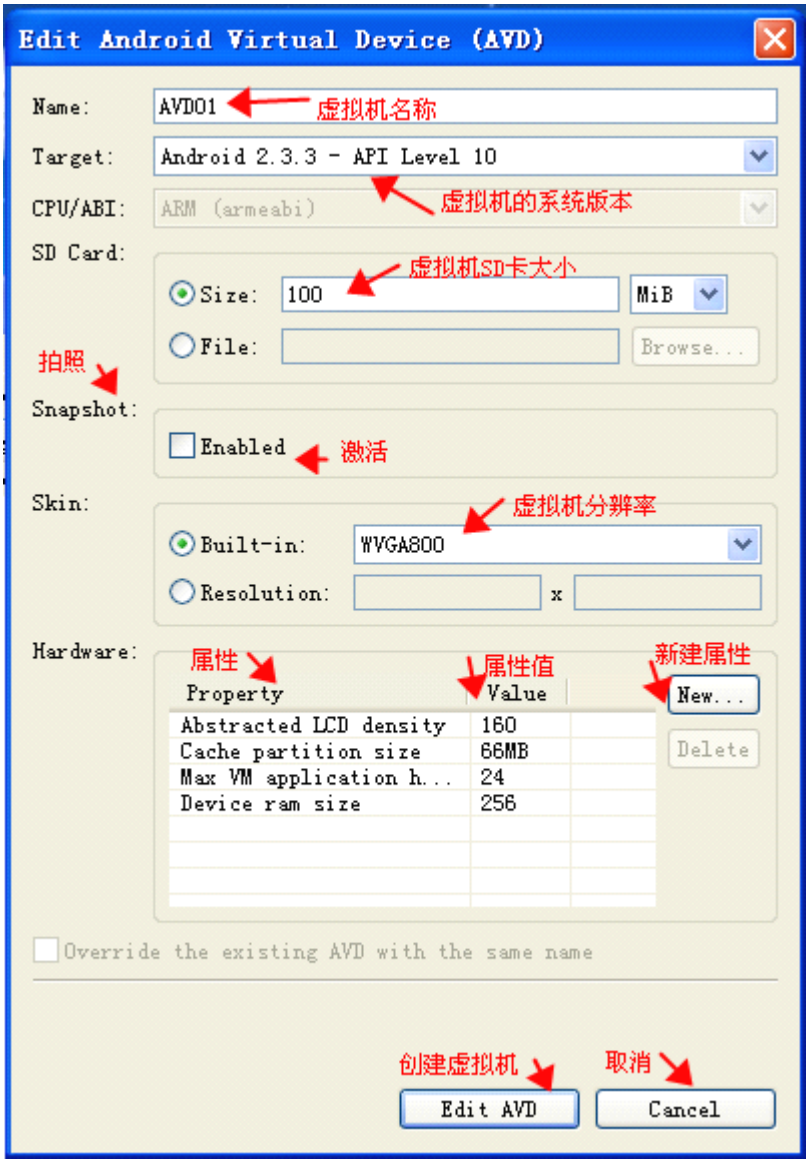
【;E:\Android\android-sdk-windows\tools;E:\Android\android-sdk-windows\platform-tools】

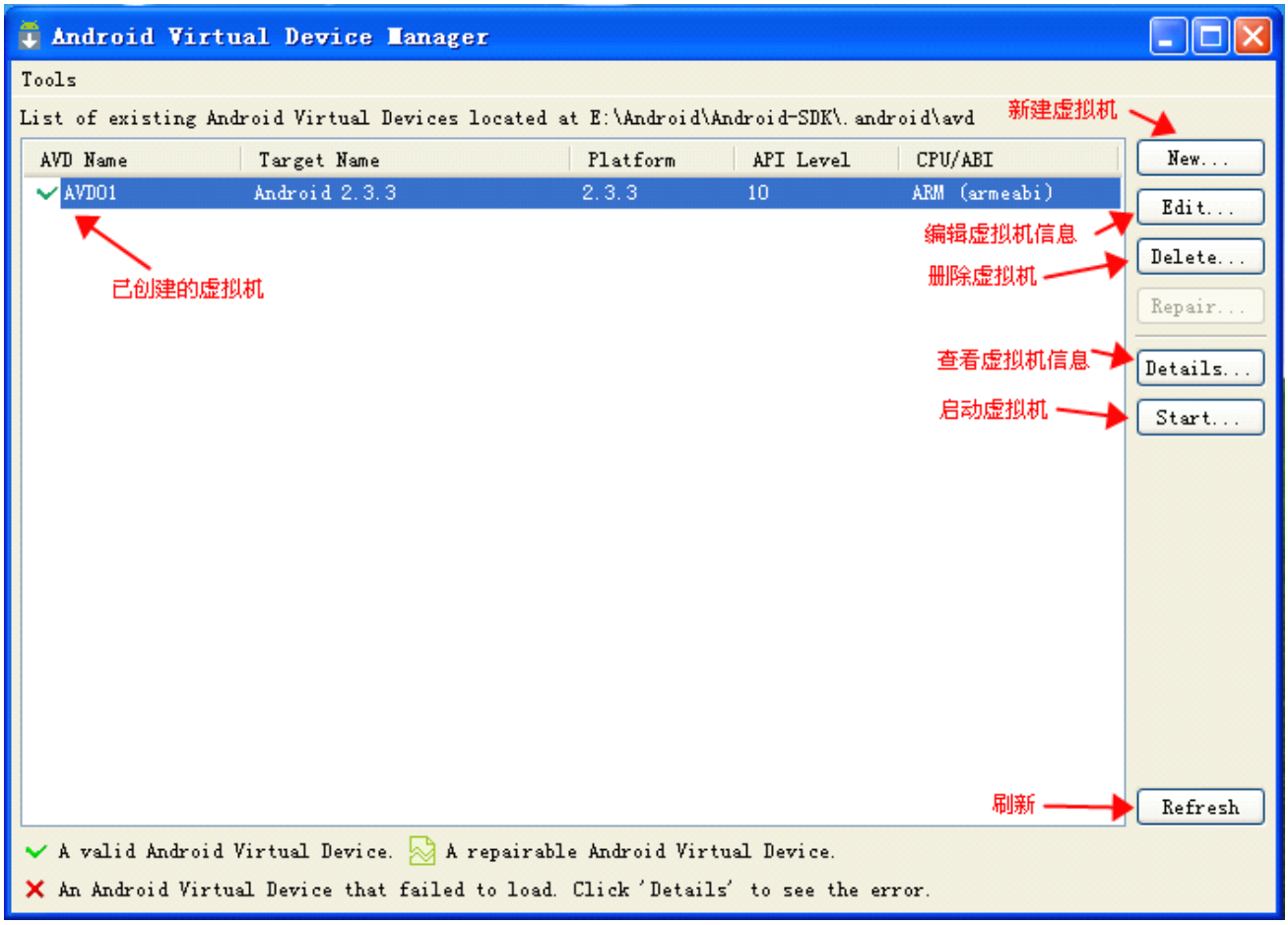
其中的分号不能少，分号在这里是起到分隔的作用。

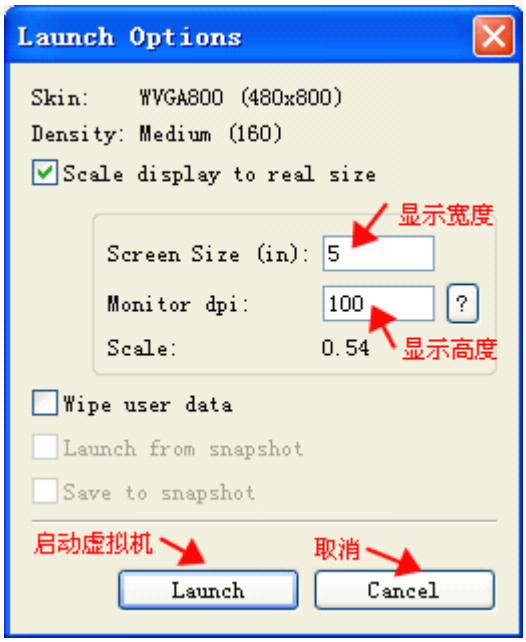
配置完成之后，分别点击【开始】--【运行】--输入【cmd】--输入【android -h】--按【回车键】，若看到以下信息，则代表配置成功。



1.2.5 配置 ADV 安卓虚拟机 并写 hello android 测试



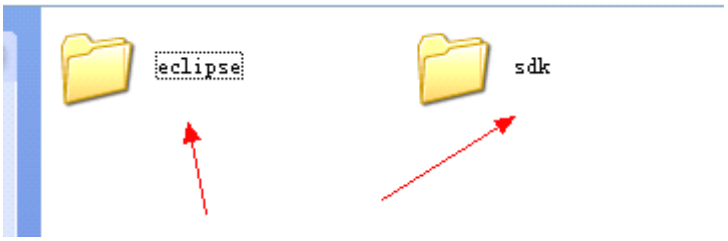


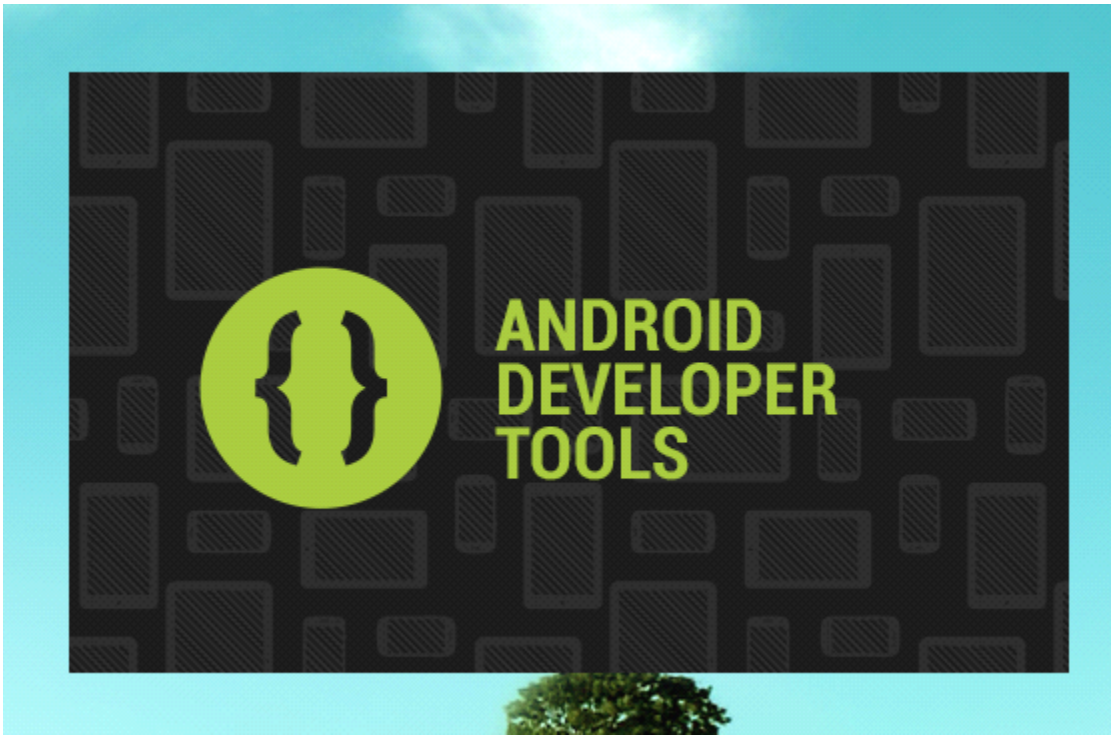


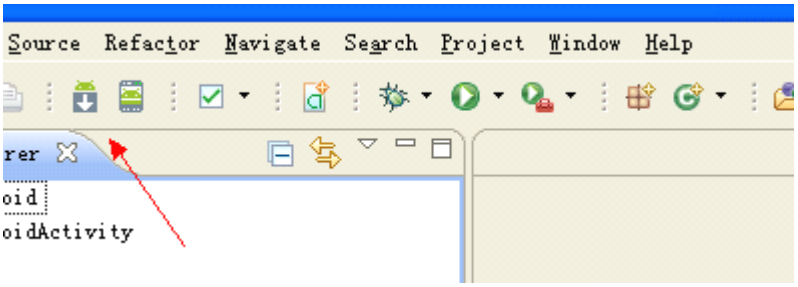
1.2.6 特别赠送:直接下载 Android Developer Tools 即可省略以上 5 个步骤

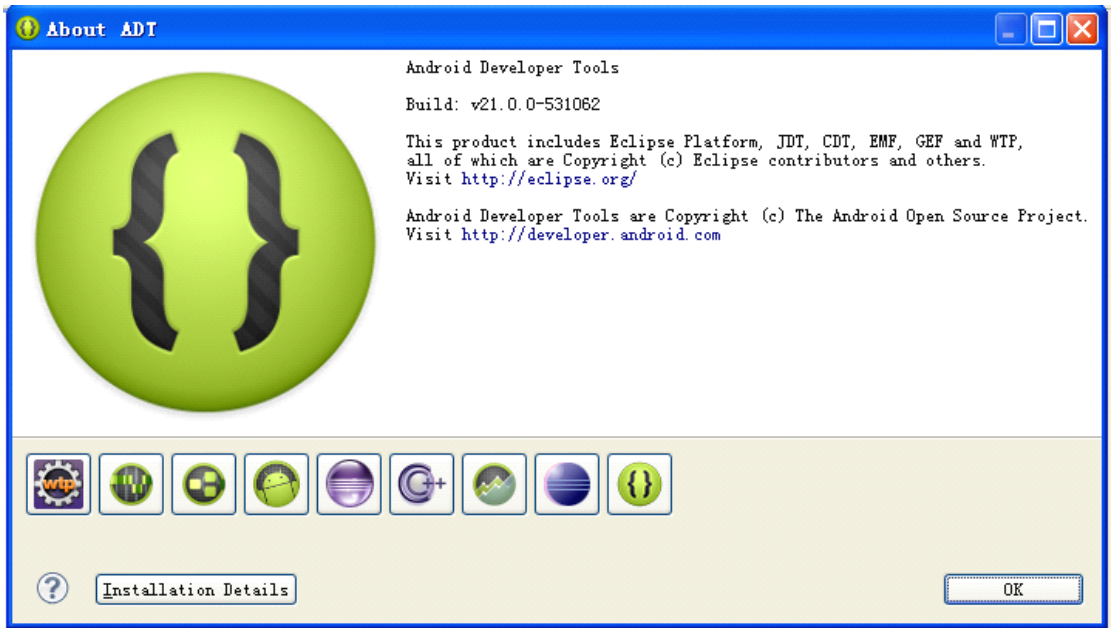
直接下载 Android Developer Tools 即可使用
官方已经推出集成 ADT 和最新 SDK(4.2 版本)的 eclipse 了
命名为 Android Developer Tools
下载地址:http://dl.google.com/android/adt/adt-bundle-windows-x86.zip
或者联系作者获取

如需安装其他版本 SDK 请看第 3 步

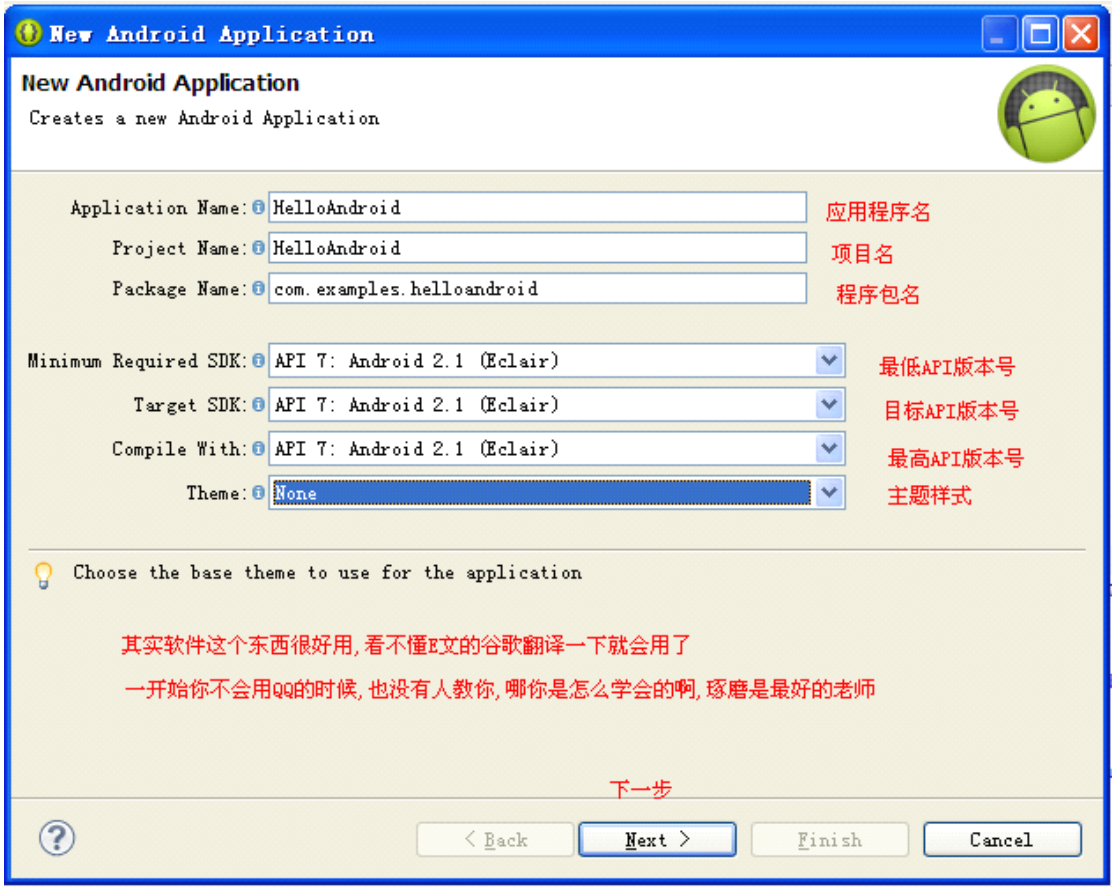


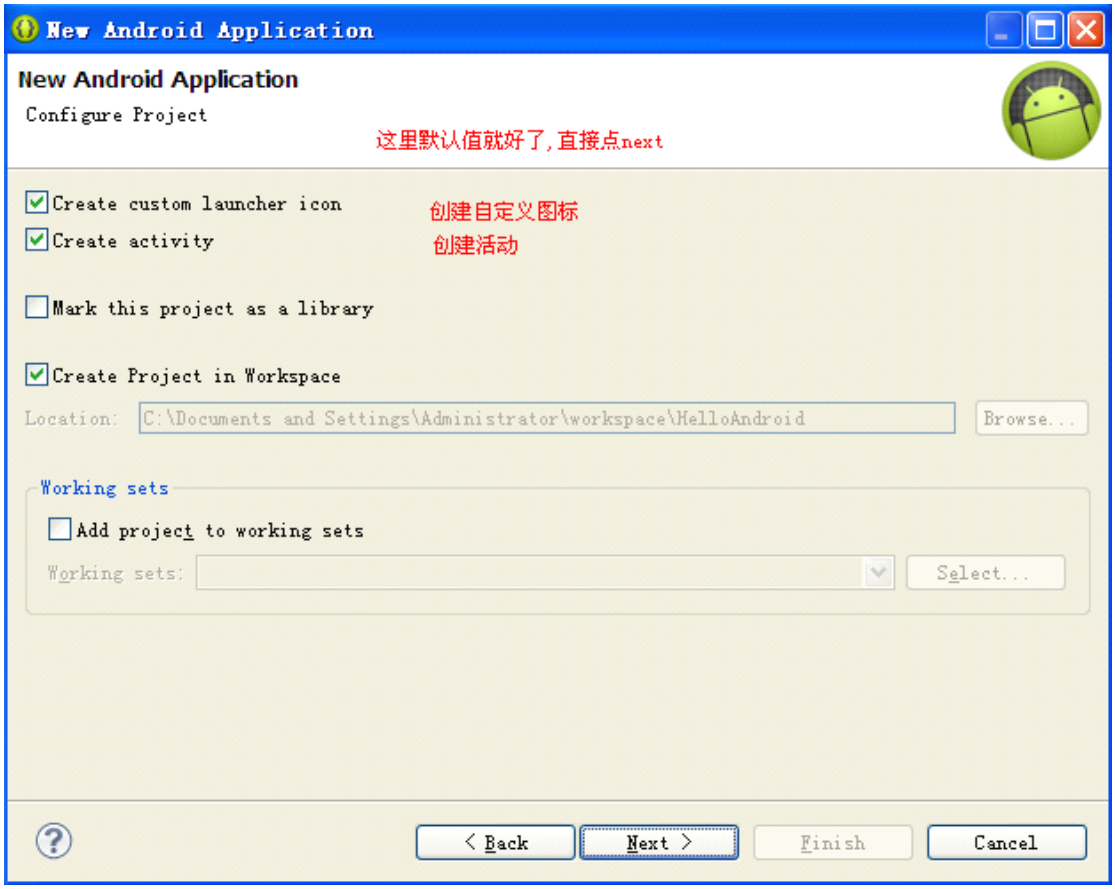


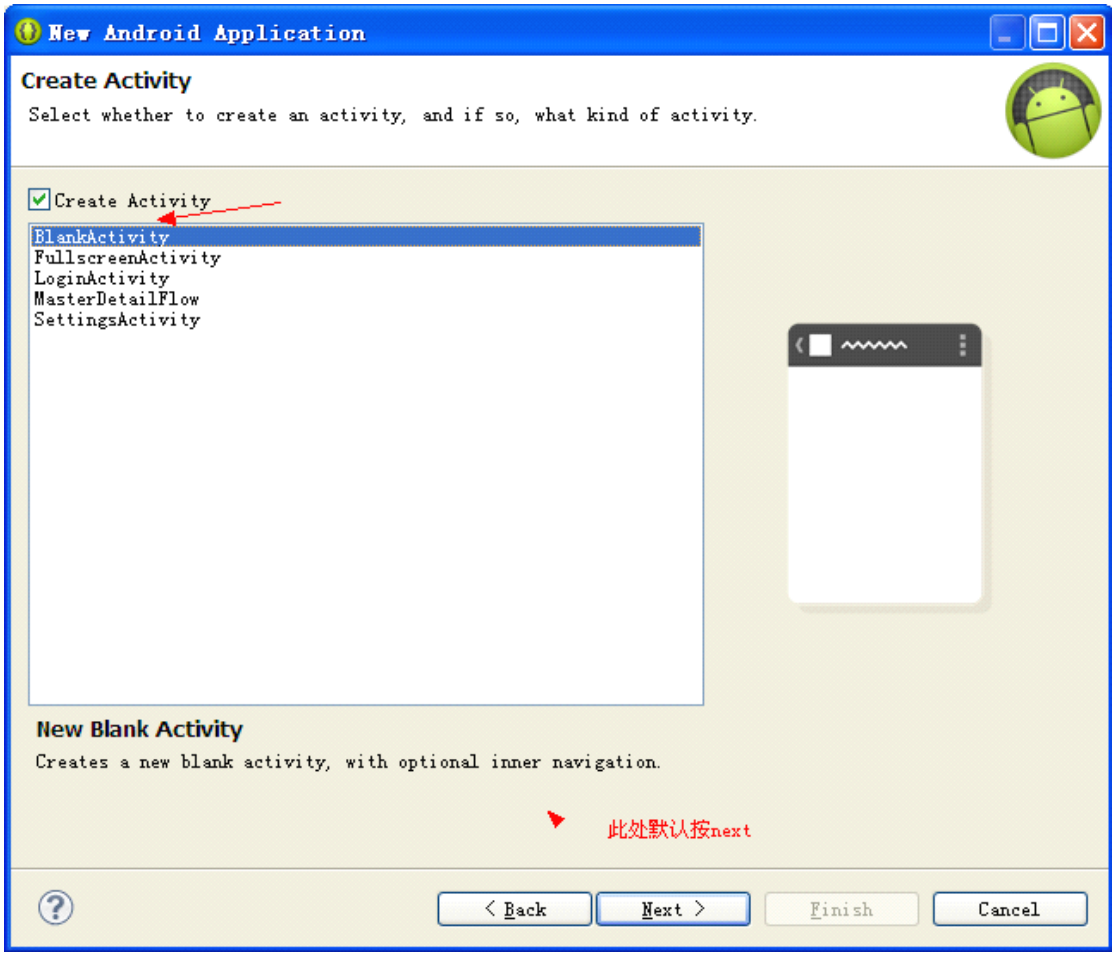


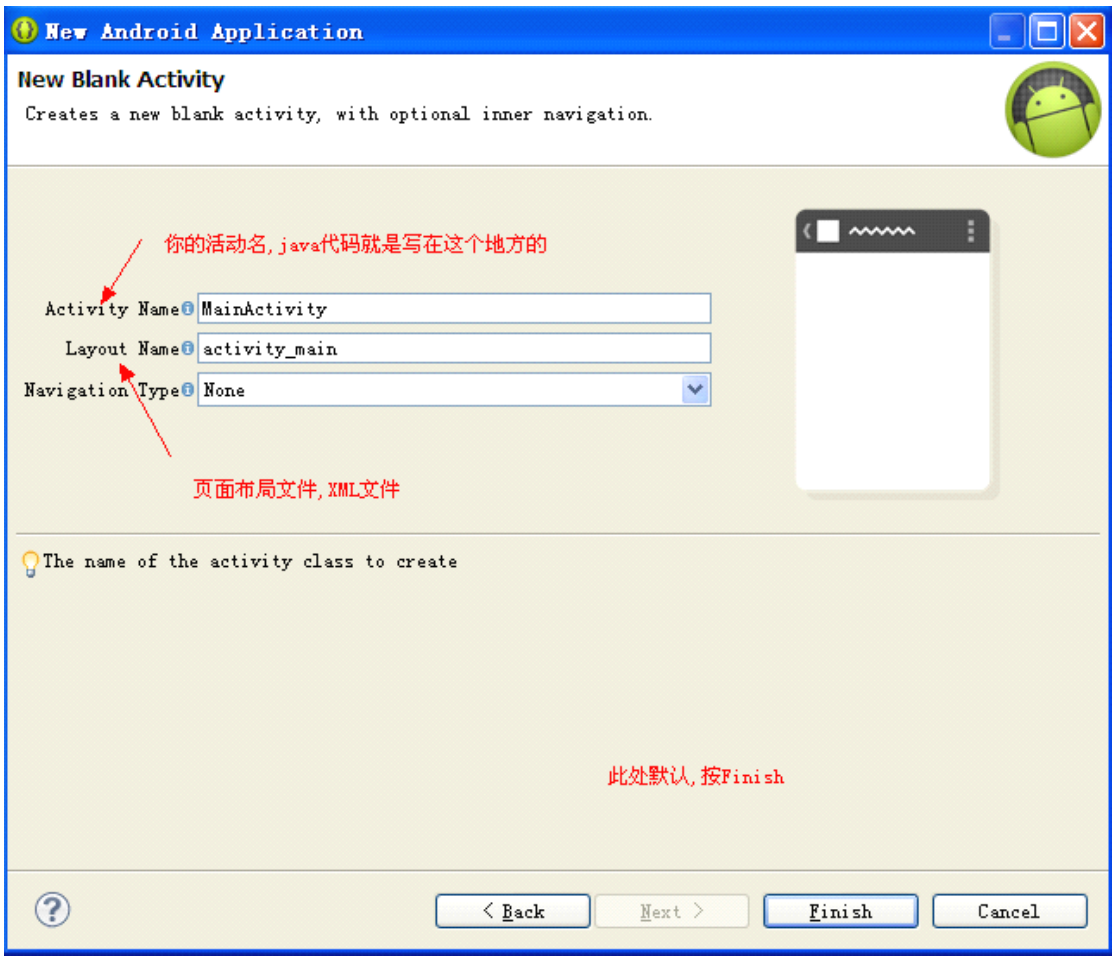


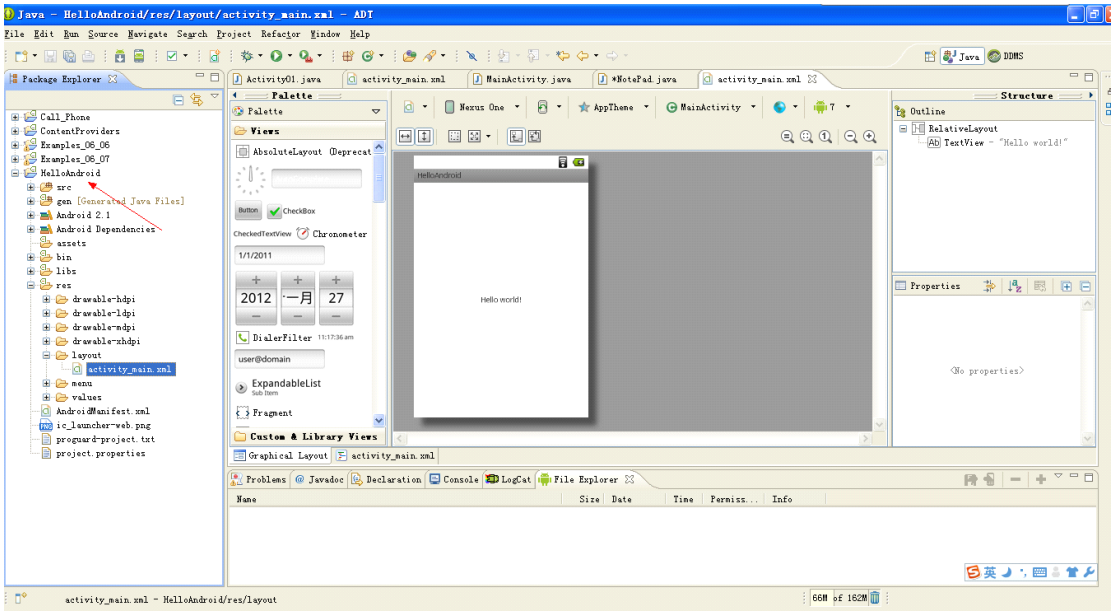
写第一个 android 程序



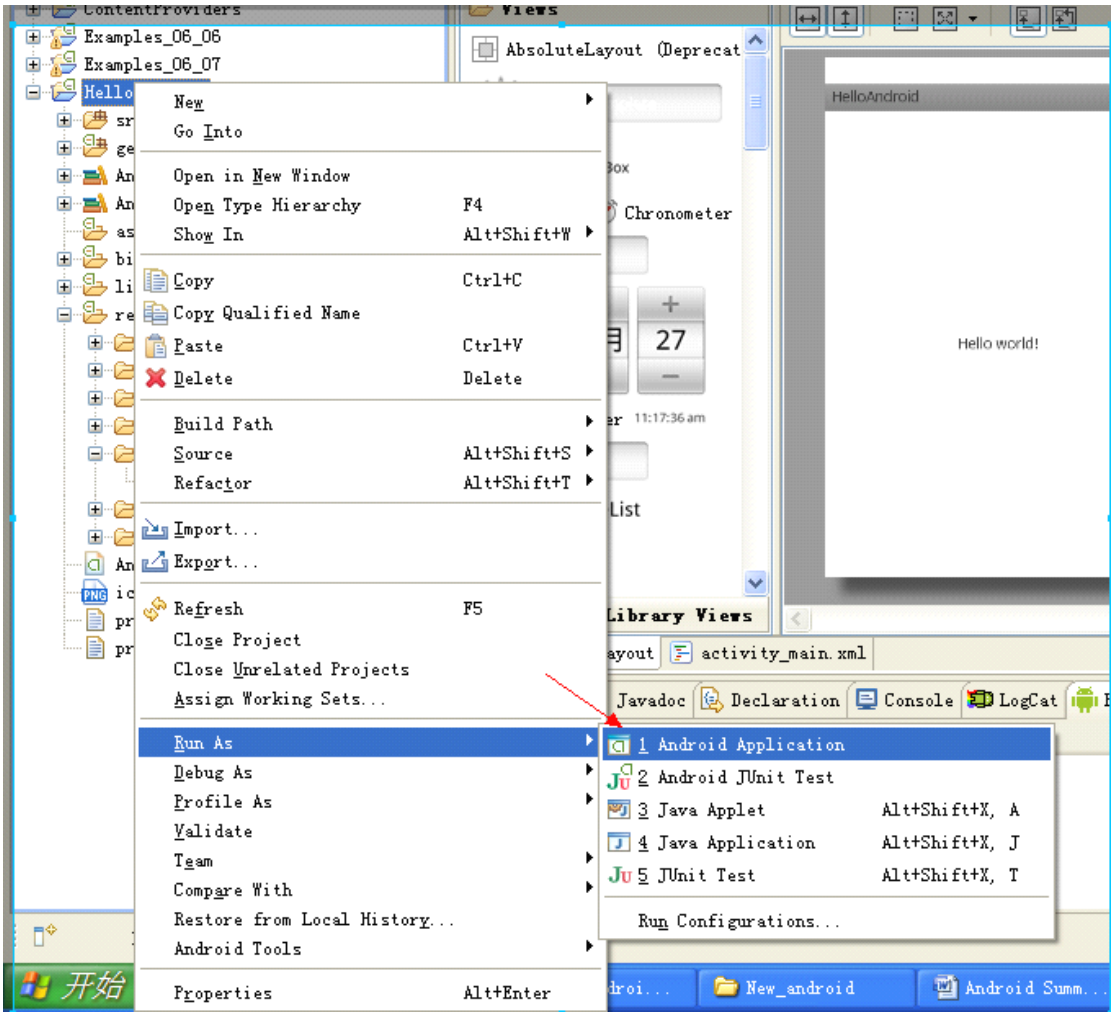








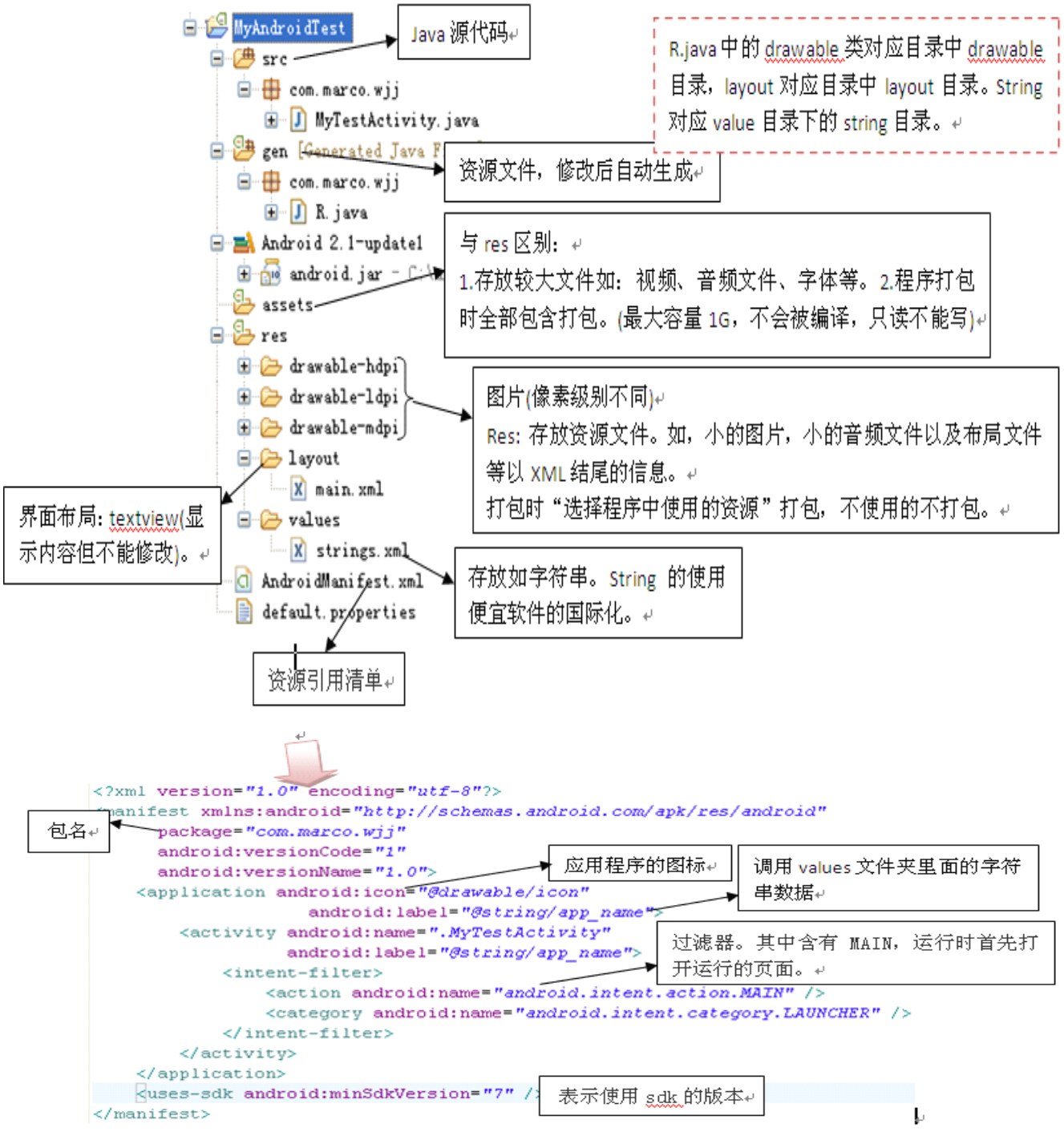
第一个程序就创建好了,如下图启动你的程序



如图,测试成功.第一天的 android 课程结束.

第二天 Android 程序设计基础

2.1 深入了解安卓




```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.marco.wjj"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        <activity android:name=".MyTestActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

2.1.1 工程结构解析

src： Java 源代码目录(只允许有一个包)
AndroidMainfest.xml (清单文件，描述应用程序构成、组件、权限等配置信息)
gen/R.java 资源文件，修改后自动重新生成
Android 库
assets: 静态资产文件(用于存放不编译的资源最大支持 1G)
drawable:程序图标(ico.png)
layout: 布局文件夹
values:程序用到的 string、颜色(string.xml)
classes.dex 编译的 java 二进制码 Android 平台上的可执行文件
ActivityLifeCycle.apk Android 安装包(APK 包)
resources.ap_ 资源信息文件

2.1.2 Android 中 JAVA 包功能描述

- **android.app** ： 提供高层程序模型、提供基本的运行环境
- **android.content** ： 对设备上的数据进行访问和发布的类
- **android.database** ： 通过内容提供者浏览和操作数据库
- **android.graphics** ： 底层的图形库
- **android.location** ： 定位和相关服务的类
- **android.media** ： 管理多种音频、视频的媒体接口
- **android.net** ： 提供帮助网络访问的类
- **android.os** ： 提供了系统服务、消息传输、**IPC** 机制
- **android.opengl** ： 提供 **OpenGL** 的工具
- **android.provider** ： 提供类访问 **Android** 内容提供者
- **android.telephony** ： 提供与拨打电话相关的 **API** 交互
- **android.view** ： 提供基础的用户界面接口框架
- **android.util** ： 涉及工具性的方法，例如时间日期的操作
- **android.webkit** ： 默认浏览器操作接口
- **android.widget** ： 包含各种 **UI** 元素在应用程序的屏幕中使用

2.1.3 Android 程序核心组件

View:	界面视图、组织 UI 控件
Intent:	意图,支持组件间通信
Activity:	处理界面与 UI 互动
Content Provider:	存储共享数据
IntentReceiver:	接收信息及事件处理
Service:	后台服务(如硬件与驱动的服务)
Notification:	消息与通知

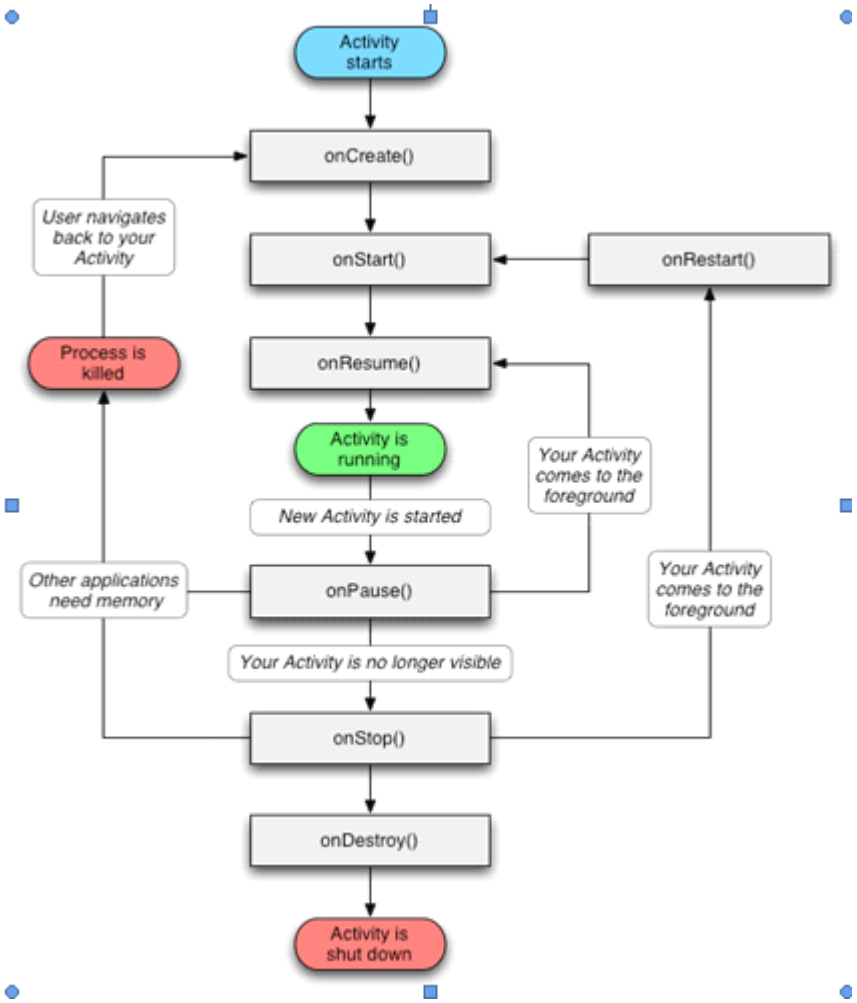
2.2 了解 Activity

2.2.1 Activity 的概念

- 活动是最基本的 **Android** 应用程序组件
- 一个活动通常就是一个单独的屏幕,它将会显示由视图控件组成的用户接口，并对事件做出响应以启动其他组件。
- 当一个新的屏幕打开后，前一个屏幕将会暂停，并保存在历史堆栈中。用户可以返回到历史堆栈中的前一个屏幕。当屏幕不再使用时，还可以从历史堆栈中删除。**Android** 将会保留从主屏幕到每一个应用的运行屏幕。
- 所有应用的 **Activity** 都继承于 **android.app.Activity** 类

2.2.2 Activity 的生命周期

```
void onCreate(Bundle savedInstanceState)
void onStart()
void onRestart()
void onResume()
void onPause()
void onStop()
void onDestroy()
```



Activity 有三个状态：

- 当它在屏幕前台时（位于当前任务堆栈的顶部），它是激活或运行状态。它就是响应用户操作的 **Activity**。
- 当它失去焦点但仍然对用户可见时（如右图），它处于暂停状态。即在它之上有另外一个 **Activity**。这个 **Activity** 也许是透明的，或者没有完全覆盖全屏，所以被暂停的 **Activity** 仍对用户可见。暂停的 **Activity** 仍然是存活状态（它保留着所有的状态和成员信息并保持和窗口管理器的连接），但系统处于极低内存时仍然可以杀死这个 **Activity**。
- 完全被另一个 **Activity** 覆盖时则处于停止状态。它仍然保留所有的状态和成员信息。然而对用户是不可见的，所以它的窗口将被隐藏，如果其它地方需要内存，则系统经常会杀死这个 **Activity**。

当 **Activity** 从一种状态转变到另一种状态时，会调用以下保护方法来通知这种变化：

这七个方法定义了 **Activity** 的完整生命周期。实现这些方法可以帮助我们监视其中的三个嵌套生命周期循环：

- **Activity** 的完整生命周期自第一次调用 **onCreate()**开始，直至调用 **onDestroy()**为止。**Activity** 在 **onCreate()**中设置所有“全局”状态以完成初始化，而在 **onDestroy()**中

释放所有系统资源。例如，如果 Activity 有一个线程在后台运行从网络上下载数据，它会在 onCreate()创建线程，而在 onDestroy()销毁线程。

- Activity 的可视生命周期自 onStart()调用开始直到相应的 onStop()调用结束。在此期间，用户可以在屏幕上看到 Activity，尽管它也许并不是位于前台或者也不与用户进行交互。在这两个方法之间，我们可以保留用来向用户显示这个 Activity 所需的资源。例如，当用户不再看见我们显示的内容时，我们可以在 onStart()中注册一个 BroadcastReceiver 来监控会影响 UI 的变化，而在 onStop()中来注消。onStart() 和 onStop() 方法可以随着应用程序是否为用户可见而被多次调用。
- Activity 的前台生命周期自 onResume()调用起，至相应的 onPause()调用为止。在此期间，Activity 位于前台最上面并与用户进行交互。Activity 会经常在暂停和恢复之间进行状态转换——例如当设备转入休眠状态或者有新的 Activity 启动时，将调用 onPause() 方法。当 Activity 获得结果或者接收到新的 Intent 时会调用 onResume()方法。关于前台生命周期循环的例子请见 PPT 下方备注栏。

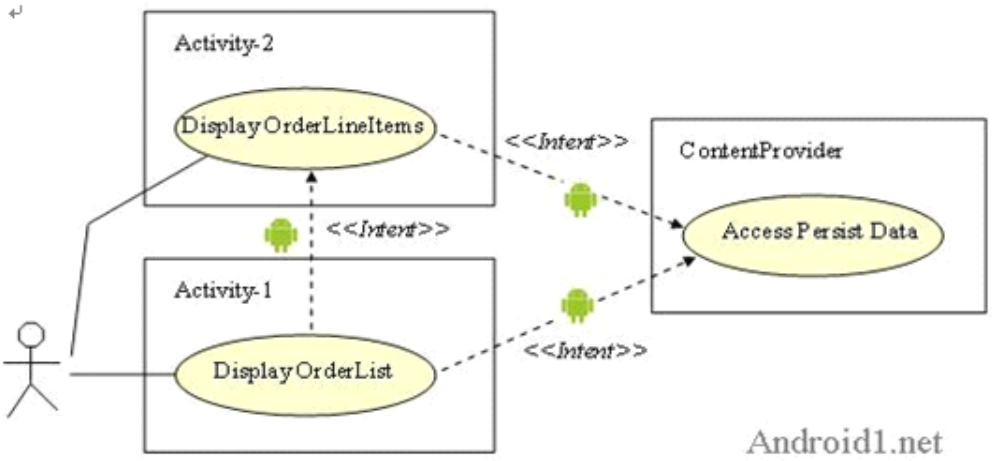
(活动通俗来说就活动的可操纵的窗口)

2.3 了解 Intent

2.3.1 什么是意图:

1.Android 基本的设计理念是鼓励减少组件间的耦合
2.Intent 提供通用的消息系统，它允许在组件与组件间传递 Intent 来执行动作和产生事件。
3.使用 Intent 可以激活 Android 应用的三个核心组件：活动、服务和广播接收器。
4.Intent 可以划分成显式意图和隐式意图。
5.显式意图：调用 Intent.setComponent()或 Intent.setClass()方法指定了组件名或类对象的 Intent 为显式意图，显式意图明确指定了 Intent 应该传递给哪个组件。
6.隐式意图：没有调用 Intent.setComponent()或 Intent.setClass()方法指定组件名或类对象的 Intent 为隐式意图。 Android 会根据 intent-filter 中设置的动作(action)、类别(category)、数据（URI 和数据类型）找到最合适的组件来处理这个意图。Intent Filter（过滤器）其实就是用来匹配隐式 Intent 的。想要接收使用 startActivity()方法传递的隐式意图的活动必须在它们的意图过滤器中包含"android.intent.category.DEFAULT"

(简单举个例子,我意图打劫,所以意图就是那么一回事)



2.4 Bundle 类的作用

Bundle类用作携带数据，它类似于Map，用于存放key-value名值对形式的值。相对于Map，它提供了各种常用类型的putXxx()/getXxx()方法，如:putString()/getString()和putInt()/getInt()，putXxx()用于往Bundle对象放入数据，getXxx()方法用于从Bundle对象里获取数据。Bundle的内部实际上是使用了HashMap<String, Object>类型的变量来存放putXxx()方法放入的值：

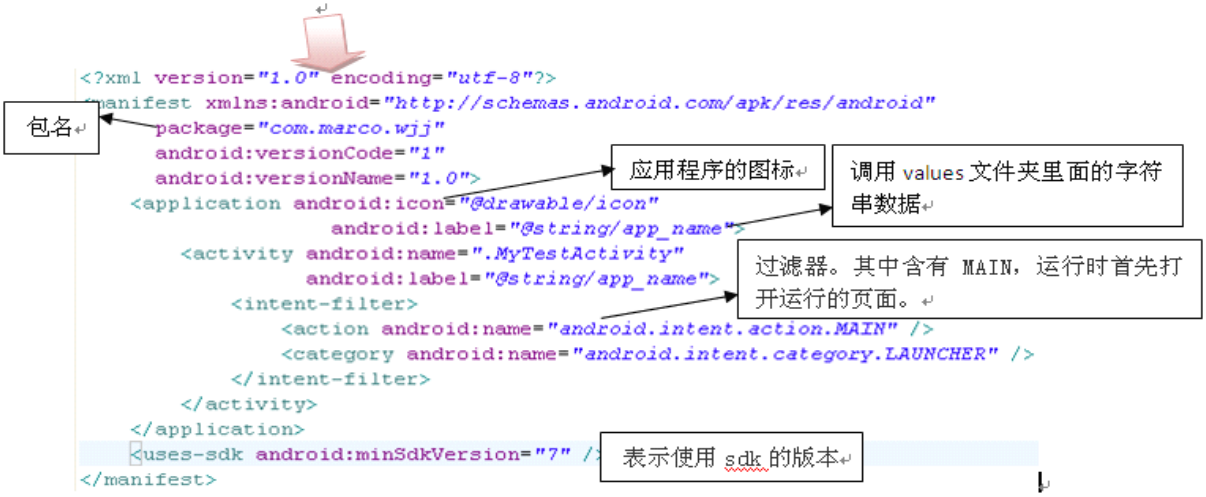
```
public final class Bundle implements Parcelable, Cloneable {
        ......
        Map<String, Object> mMap;
        public Bundle() {
            mMap = new HashMap<String, Object>();
            ......
        }
        public void putString(String key, String value) {
            mMap.put(key, value);
        }
        public String getString(String key) {
            Object o = mMap.get(key);
            return (String) o;
            ........//类型转换失败后会返回null，这里省略了类型转换失败后的处理代码
        }
}
```

在调用Bundle对象的getXxx()方法时，方法内部会从该变量中获取数据，然后对数据进行类型转换，转换成什么类型由方法的Xxx决定，getXxx()方法会把转换后的值返回。

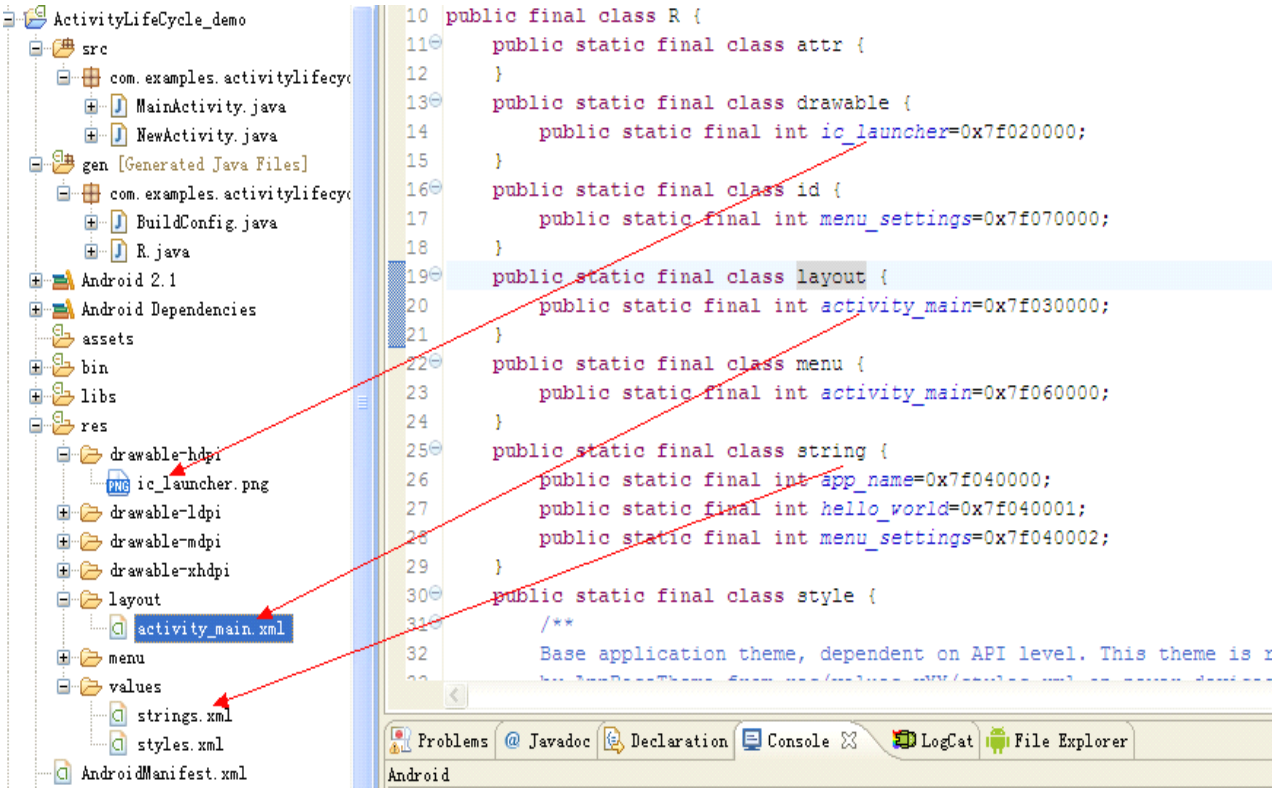
2.5 回顾 helloandroid 看看安卓程序是怎么运作的

为什么到了这里我们才讲 android 应用是怎么运作的呢?到了这里我们已经初步了解 activity,和 intent,和 Bundle 了!

首先我们知道 java 是从 Main 进入程序的,那么 android 是怎样进入程序的呢?细心的朋友或许已经知道了, AndroidManifest.xml,就是我们程序的入口了.在 2.1 节里面,有这样的一幅图



包名是我们创建目录的包名,包名下面是版本号,
<application>标签里注册了图标,标题名,还有 activity,　@是标准格式,在 R 里边会有唯一的一个 ID,比如 XML 是用来布局的,那么在 R 里则会生成一个对应的 ID,
<intent-filter>标签监听器,监听 activity 第一个打开



那么我们就可以去看看 MainActivity 了

```java
package com.examples.activitylifecycle_demo;

import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

首先我们看看是不是有 activity 的 onCreate 啊!上面说,不论如何都要先执行它,然后传进来一个 Bundle 类型的值,然后又调用自身这个类!底层代码已经帮我们实现好了,我们只需要用就行了.然后 setContentView 是得到内容视图,得到那个视图呢?刚刚我们不是说了吗,注册一个 XML 就会有一个 R 值.获取到 activity_main 这个视图
那么我们来看看布局文件 xml 里是什么,

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

里面就一个文本框,字符串是找 strings 里边的一个字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">ActivityLifeCycle_demo</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>

</resources>
```

所以我们创建 helloandroid 的时候,看到的就是 hello world 了!
在此我们了解完了 android 的运作过程,那么我们来写第二个程序吧!

2.6 写第二个程序 ActivityLifeCycle

按照创建 helloandroid 程序的方法创建一个 activityLifeCycle 的项目
然后点击包右键新建一个类

为应用添加新的 Activity

第一步：新建一个继承Activity的类，如：NewActivity

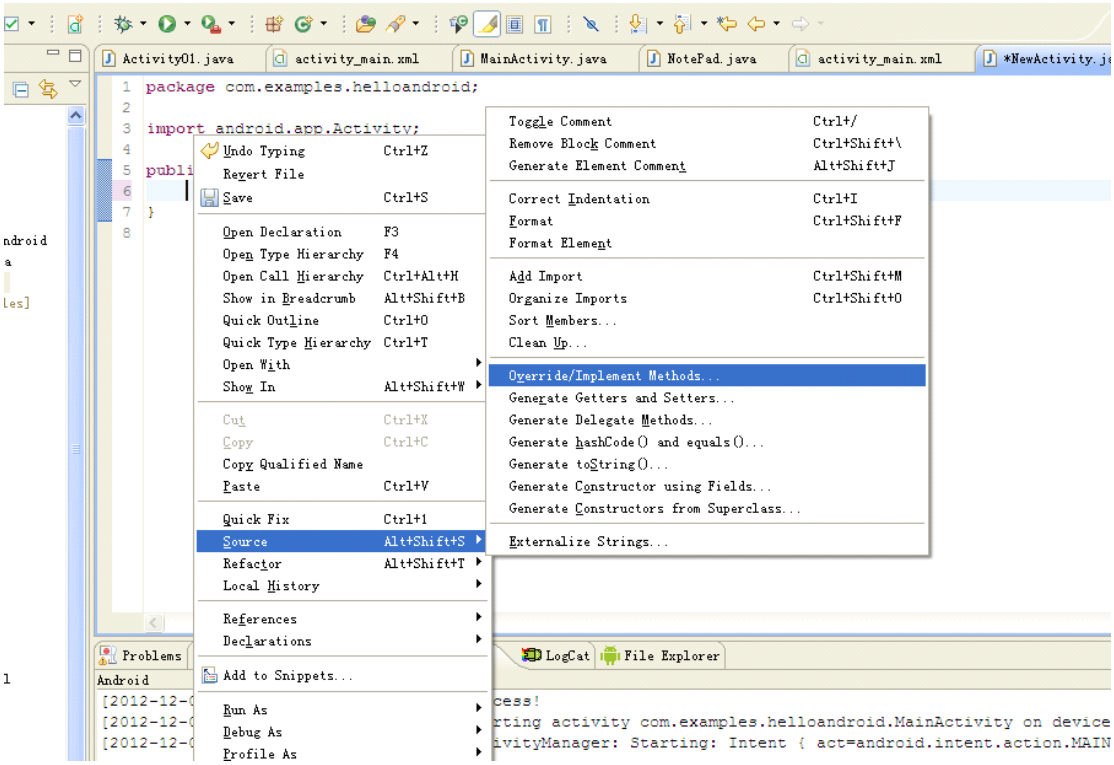
```
public class NewActivity extends Activity {
  @Override protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
            //这里可以使用setContentView(R.layout.xxx)显示某个视图....
  }
}
```

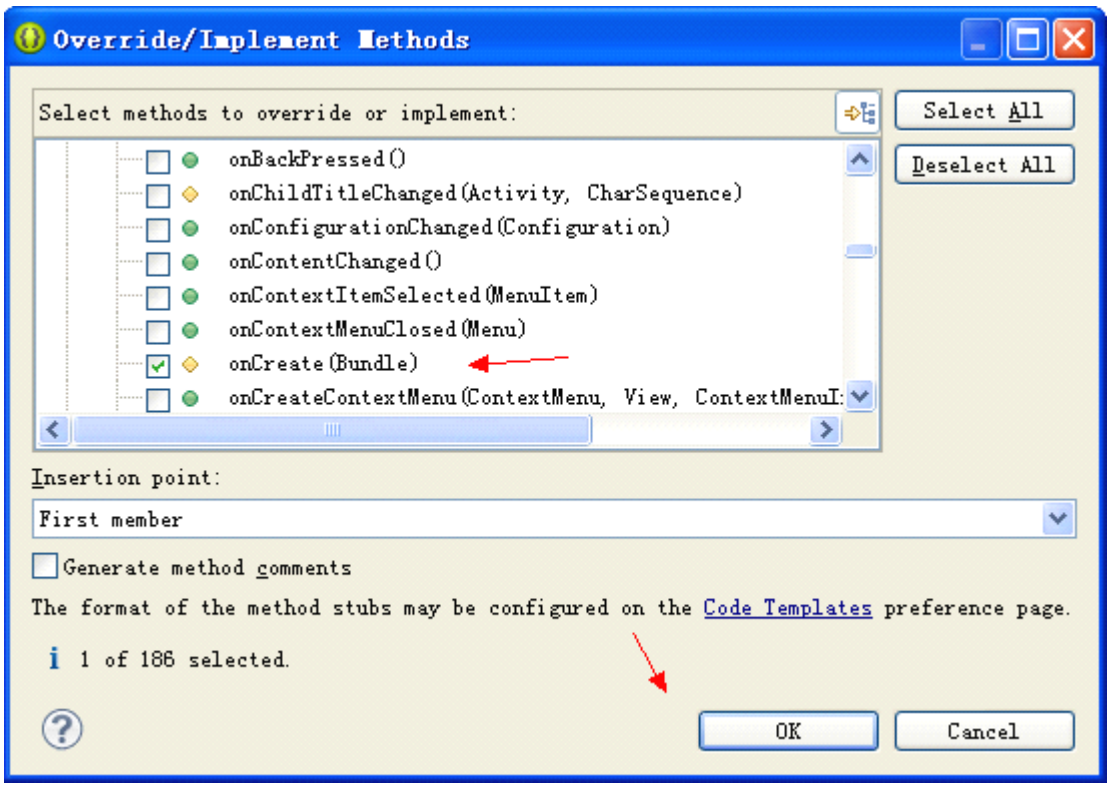
第二步：需要在功能清单AndroidManifest.xml文件中添加进上面Activity配置代码(红色部分)：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lxt008"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    .....
            <activity android:name=".NewActivity" android:label="新activity的页面标题"/>
  </application>
  ...
</manifest>
```

android:name属性值的前面加了一个点表示NewActivity是当前包com.lxt008下的类，如果类在应用的当前包下，可以省略点符号，如果类在应用的子包下必须加点，如：NewActivity类在com.lxt008.user包下可以这样写：<activity android:name=".user.NewActivity" />

这里说明一下 onCreate 是怎么弄出来的!





在前面打勾按下 OK

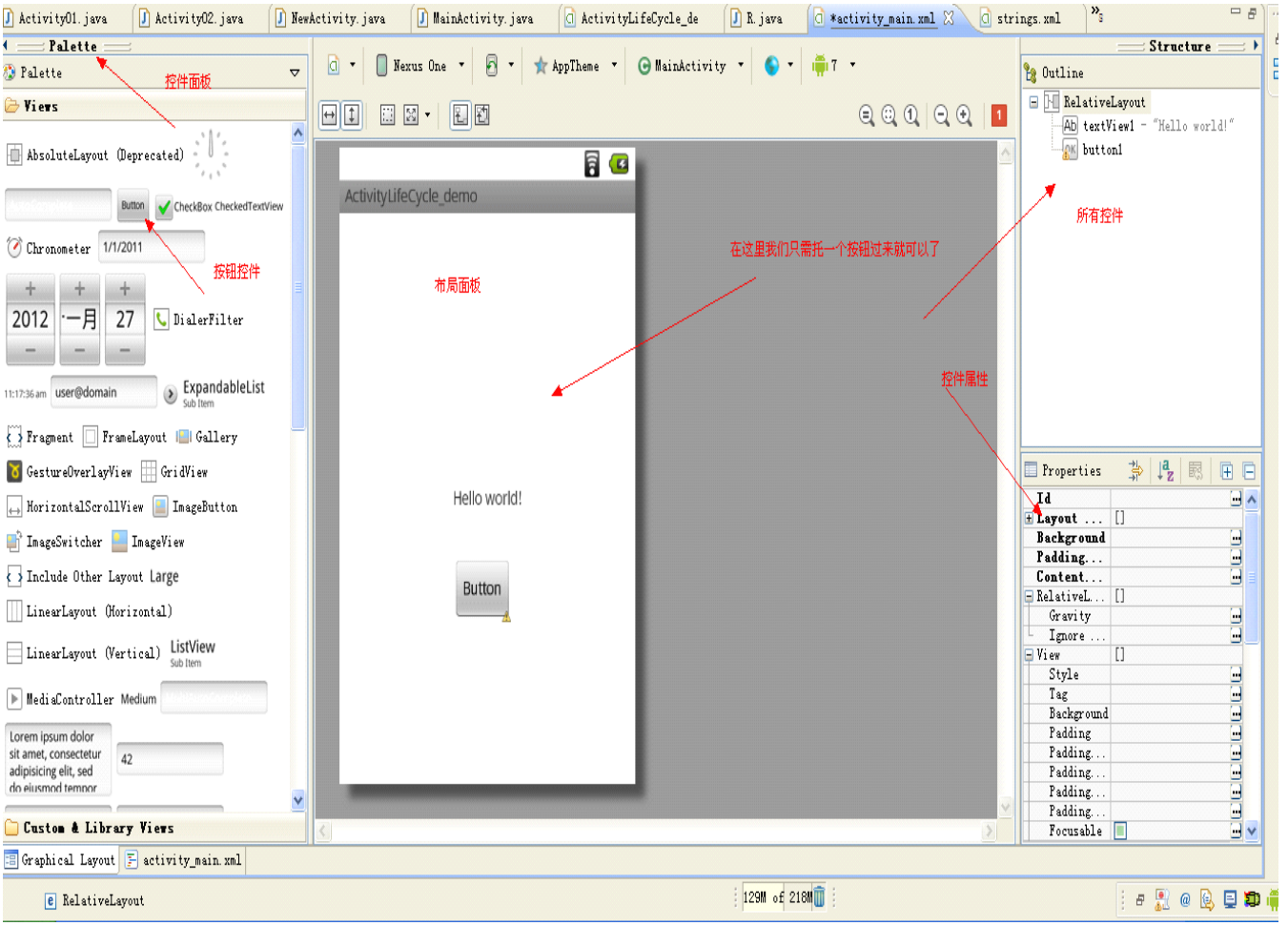
打开新的 **Activity** ， 不传递参数

在一个Activity中可以使用系统提供的startActivity(Intent intent)方法打开新的Activity，在打开新的Activity前，你可以决定是否为新的Activity传递参数：

```
第一种：打开新的Activity，不传递参数
public class MainActivity extends Activity {
 @Override protected void onCreate(Bundle savedInstanceState) {
            .....
            Button button =(Button) this.findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener(){
                        //点击该按钮会打开一个新的Activity
                        public void onClick(View v) {
                        //新建一个显式意图，第一个参数为当前Activity类对象
                        //第二个参数为你要打开的Activity类
                        startActivity(new Intent(MainActivity.this, NewActivity.class));
            }});
      }
}
```

上面定义了一个按钮,findViewById 是根据 ID 找到视图,R.id.button 那么你的 XML 是不是要添加一个按钮啊?在最新版的 ADT21 中,我们可以轻松的布局了,在 ADT8 中拖动控件还是很麻烦的

学过 java 的都会问,为什么不直接 NEW 一个按钮啊,那么麻烦,android 这样是为了降低耦合度,高内聚,低耦合.



当让你也可以再 xml 里自己写控件

我们再回过头来看 button. setOnClickListener(new `OnClickListener(){ }`)

这里的 setOnClickListener 是放置一个点击监听器里边有 new 了一个内部类

为什么要这样写?当让你也可以不这样写,这样写是为了方便,学过 java 的应该都知道吧!你另外写一个类,然后再调用它也可以.

实现 `OnClickListener` 接口自动实现 `onClick` 方法

```
//新建一个显式意图，第一个参数为当前Activity类对象
//第二个参数为你要打开的Activity类
startActivity(new Intent(MainActivity.this, NewActivity.class));
```

这句可以分开来写

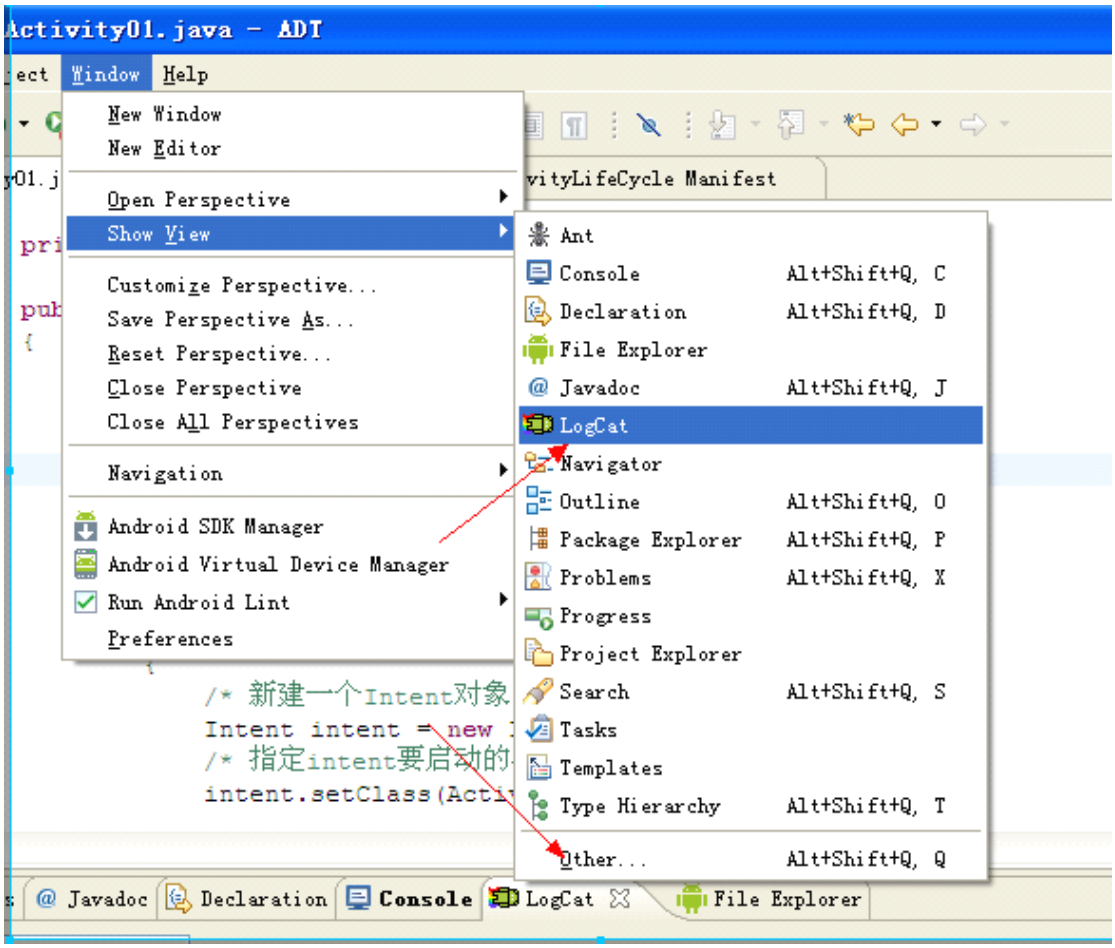
```
Intent intent=new Intent();
intent.setClass(MainActivity.this, NewActivity.class);
```

```
startActivity(intent);
```

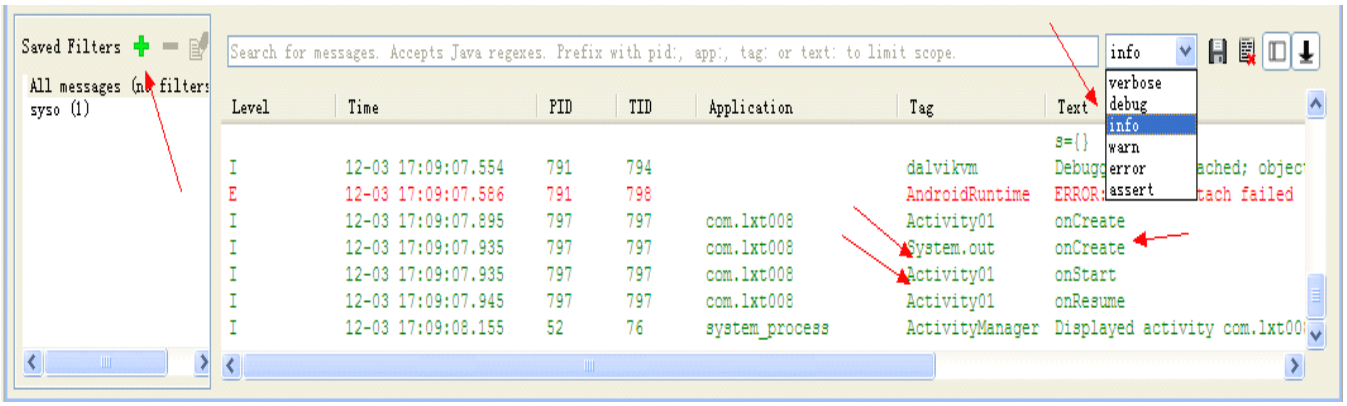
刚刚我们学了意图,意图是活动之间传递信息的信使,第一个参数是自身的一个活动,第二个是要转过去的活动.

2.7 使用过滤器:

如何弄出 LoCat



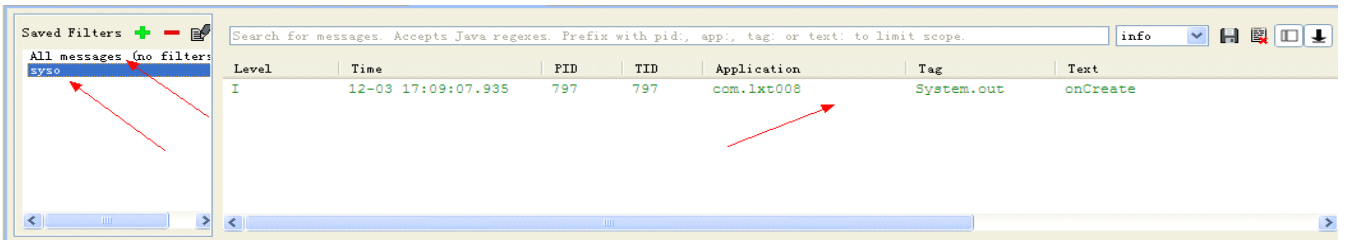
快捷栏里没有的话到 Other 里边找

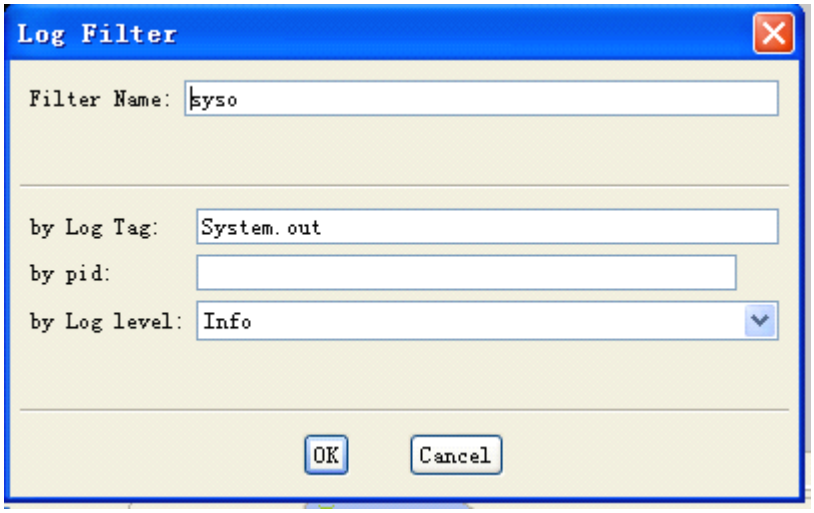


相信学过 java 的对输出语句并不陌生,点击上面绿色的那个加号就是添加过滤器的,by log level 是日志类型,如上图右边的箭头,从上到下有所有日志,捉虫日志,信息日志,警告日志,错误日志…

设置 syso 过滤器

Syso 的记录日志我觉得没那么乱,容易看出来,单独显示出来的,all 是所有信息





(详情请参考 demo, ActivityLifeCycle)

打开新的 Activity，传递参数给它

```
第二种：打开新的Activity，并传递若干个参数给它：
public class MainActivity extends Activity {
  @Override protected void onCreate(Bundle savedInstanceState) {
            .....
             button.setOnClickListener(new View.OnClickListener(){
                 public void onClick(View v) {
                   Intent intent = new Intent(MainActivity.this, NewActivity.class)
                   Bundle bundle = new Bundle();//该类用作携带数据
                   bundle.putString("name", "lxt008");
                   bundle.putInt("age", 80);
                   intent.putExtras(bundle);//附带上额外的数据
                   startActivity(intent);
            }});}
}
在新的Activity中接收前面Activity传递过来的参数：
public class NewActivity extends Activity {
        @Override protected void onCreate(Bundle savedInstanceState) {
                .....
                Bundle bundle = this.getIntent().getExtras();
                String name = bundle.getString("name");
                int age = bundle.getInt("age");
        }}
```

得到新 Activity 关闭后返回的数据

如果你想在Activity中得到新打开Activity关闭后返回的数据，你需要使用系统提供的startActivityForResult(Intent intent, int requestCode)方法打开新的Activity，新的Activity关闭后会向前面的Activity传回数据，为了得到传回的数据，你必须在前面的Activity中重写onActivityResult(int requestCode, int resultCode, Intent data)方法：

```
public class MainActivity  extends Activity {
    @Override protected void onCreate(Bundle savedInstanceState) {
            .....
            Button button =(Button) this.findViewById(R.id.button);
               button.setOnClickListener(new View.OnClickListener(){//点击该按钮会打开一个新的Activity
                        public void onClick(View v) {
                        //第二个参数为请求码，可以根据业务需求自己编号
                        startActivityForResult (new Intent(MainActivity.this, NewActivity.class), 1);
            }});
    }
  //第一个参数为请求码，即调用startActivityForResult()传递过去的值
  //第二个参数为结果码，结果码用于标识返回数据来自哪个新Activity
  @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
            String result = data.getExtras().getString("result");//得到新Activity 关闭后返回的数据
  }
}
```

当新Activity关闭后，新Activity返回的数据通过Intent进行传递，android平台会调用前面Activity的onActivityResult()方法，把存放了返回数据的Intent作为第三个输入参数传入，在onActivityResult()方法中使用第三个输入参数可以取出新Activity返回的数据。

使用startActivityForResult(Intent intent, int requestCode)方法打开新的Activity，新Activity关闭前需要向前面的Activity返回数据需要使用系统提供的setResult(int resultCode, Intent data)方法实现：

```
public class NewActivity extends Activity {
        @Override protected void onCreate(Bundle savedInstanceState) {
                .....
            button.setOnClickListener(new View.OnClickListener(){
                    public void onClick(View v) {
                            Intent intent = new Intent();//数据是使用Intent返回
                            intent.putExtra("result", "success ");//把返回数据存入Intent
                            NewActivity.this.setResult(RESULT_OK, intent);//设置返回数据
                            NewActivity.this.finish();//关闭Activity
                    }});
        }
}
```

setResult()方法的第一个参数值可以根据业务需要自己定义，上面代码中使用到的RESULT_OK是系统Activity类定义的一个常量，值为-1，代码片断如下：

```
public class android.app.Activity extends .....{
  public static final int RESULT_CANCELED = 0;
  public static final int RESULT_OK = -1;
  public static final int RESULT_FIRST_USER = 1;
}
```

请求码的作用

使用startActivityForResult(Intent intent, int requestCode)方法打开新的Activity，我们需要为startActivityForResult()方法传入一个请求码(第二个参数)。请求码的值是根据业务需要由自己设定，用于标识请求来源。例如：一个Activity有两个按钮，点击这两个按钮都会打开同一个Activity，不管是那个按钮打开新Activity，当这个新Activity关闭后，系统都会调用前面Activity的onActivityResult(int requestCode, int resultCode, Intent data)方法。在onActivityResult()方法如果需要知道新Activity是由那个按钮打开的，并且要做出相应的业务处理，这时可以这样做：

```
@Override  public void onCreate(Bundle savedInstanceState) {
    ....
    button1.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                    startActivityForResult (new Intent(MainActivity.this,  NewActivity.class), 1);
            }});
    button2.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                    startActivityForResult (new Intent(MainActivity.this,  NewActivity.class), 2);
            }});
   @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       switch(requestCode){
         case 1:
           //来自按钮1的请求，作相应业务处理
         case 2:
           //来自按钮2的请求，作相应业务处理
       }
    }
}
```

结果码的作用

在一个Activity中，可能会使用startActivityForResult()方法打开多个不同的Activity处理不同的业务，当这些新Activity关闭后，系统都会调用前面Activity的onActivityResult(int requestCode, int resultCode, Intent data)方法。为了知道返回的数据来自于哪个新Activity，在onActivityResult()方法中可以这样做(ResultActivity和NewActivity为要打开的新Activity)：

```
public class ResultActivity extends Activity {
    ......
    ResultActivity.this.setResult(1, intent);
    ResultActivity.this.finish();
}
public class NewActivity extends Activity {
    ......
    NewActivity.this.setResult(2, intent);
    NewActivity.this.finish();
}
public class MainActivity extends Activity { // 在该Activity会打开ResultActivity和NewActivity
    @Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch(resultCode){
          case 1:
            // ResultActivity的返回数据
          case 2:
            // NewActivity的返回数据
        }
    }
}
```

第三天.UI 事件处理与布局管理

3.1 View 与 ViewGroup

3.1.1 Android 界面元素

- **1、View：　视图组件**
- **2、Layout：　　布局组件**
- **3、Wigets：　　UI 元素**
- **4、Menus：　　菜单**

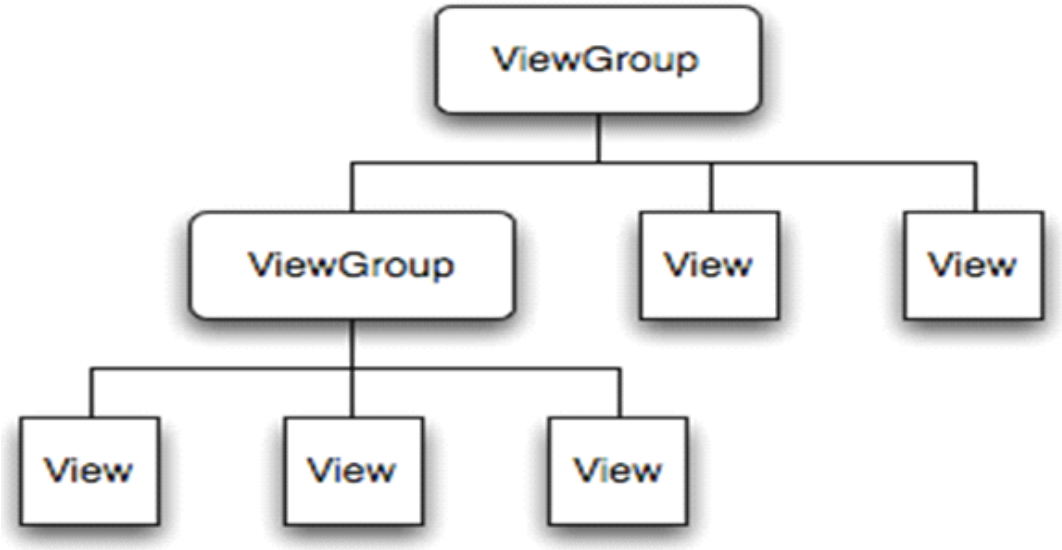
3.1.2 认识 View

1、所有高级 UI 组件都继承 View 类而实现的
2、一个 View 在屏幕上占据一块矩形区域
3、负责渲染
4、负责处理发生的事件
5、设置是否可见
6、设置是否可以获得焦点等

3.1.3 认识 ViewGroup

1、ViewGroup 对象是 android.view.ViewGroup 实例
2、 ViewGroup 是 View 的容器
3、 负责对添加进 ViewGroup 的 View 进行布局
4、一个 ViewGroup 可以加入到另一个 ViewGroup

3.1.4 View 与 ViewGroup 的关系



3.2 事件处理机制

控件事件通过设置其控件的监听器来监听并处理事件

按键按下事件：通过重写 onKeyDown 方法

按键弹起事件：通过重写 onKeyUp 方法

触笔点击事件：通过实现 onTouchEvent 方法

其他事件参考相应 UI 组件的 Demo!!

3.2.1 Toast 控件

在视图中给用户显示的短小的提示消息。

Toast.makeText(this, string, Toast.LENGTH_SHORT).show();

LENGTH_LONG：长时间显示

LENGTH_SHORT：短时间显示

3.2.2 事件处理 Demo

```
public class Activity01 extends Activity
{

	public void onCreate(Bundle savedInstanceState)
	{

		super.onCreate(savedInstanceState);

		setContentView(R.layout.main);
		//获得Button对象
		Button button_ok = (Button) findViewById(R.id.ok);
		//设置Button控件监听器
		button_ok.setOnClickListener(new Button.OnClickListener() {
			public void onClick(View v)
			{
				//这里处理事件
				DisplayToast("点击了OK按钮");
			}
		});

	}



/*按键按下所触发的事件*/
	public boolean onKeyDown(int keyCode, KeyEvent event)
	{
		switch (keyCode)
		{
			case KeyEvent.KEYCODE_DPAD_CENTER:
				DisplayToast("按下：中键");
				break;
			case KeyEvent.KEYCODE_DPAD_UP:
				DisplayToast("按下：上方向键");
				break;
			case KeyEvent.KEYCODE_DPAD_DOWN:
				DisplayToast("按下：下方向键");
				break;
			case KeyEvent.KEYCODE_DPAD_LEFT:
				DisplayToast("按下：左方向键");
				break;
			case KeyEvent.KEYCODE_DPAD_RIGHT:
				DisplayToast("按下：右方向键");
				break;
		}
		return super.onKeyDown(keyCode, event);
	}
```

```
    /* 按键弹起所触发的事件 */
        public boolean onKeyUp(int keyCode, KeyEvent event) {
            switch (keyCode) {
                case KeyEvent.KEYCODE_DPAD_CENTER:
                    DisplayToast("弹起：中键");
                    break;
                case KeyEvent.KEYCODE_DPAD_UP:
                    DisplayToast("弹起：上方向键");
                    break;
                case KeyEvent.KEYCODE_DPAD_DOWN:
                    DisplayToast("弹起：下方向键");
                    break;
                case KeyEvent.KEYCODE_DPAD_LEFT:
                    DisplayToast("弹起：左方向键");
                    break;
                case KeyEvent.KEYCODE_DPAD_RIGHT:
                    DisplayToast("弹起：右方向键");
                    break;
            }

            return super.onKeyUp(keyCode, event);
        }

        public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event) {

            return super.onKeyMultiple(keyCode, repeatCount, event);
        }
```

```
    /* 触笔事件 */
      public boolean onTouchEvent(MotionEvent event)
      {
        int iAction = event.getAction();
            if (iAction == MotionEvent.ACTION_CANCEL ||
                iAction == MotionEvent.ACTION_DOWN ||
                iAction == MotionEvent.ACTION_MOVE)
            {
                return false;
            }
            //得到触笔点击的位置
            int x = (int) event.getX();
            int y = (int) event.getY();

            DisplayToast("触笔点击坐标：("+Integer.toString(x)+","+Integer.toString(y)+")");

            return super.onTouchEvent(event);
        }

        /* 显示Toast */
        public void DisplayToast(String str)
        {
            Toast.makeText(this, str, Toast.LENGTH_SHORT).show();
        }
    }
```

详情请参考 DEMO

3.3 布界面布局方式

LinearLayout（线性布局）
AbsoluteLayout（绝对布局）
RelativeLayout（相对布局）
TableLayout（表格布局）
FrameLayout（框架布局）

3.3.1 LinearLayout（线性布局）

是常用的布局之一

一个组件一行的形式显示出来

分垂直（vertical）与水平（horizontal）两种。

main.xml：　vertical

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"
android:layout_width="fill_parent"  android:layout_height="fill_parent"  >
        <TextView
          android:text="第一行"  android:gravity="center_vertical"          android:textSize="15pt"
          android:background="#aa0000"          android:layout_width="fill_parent"
          android:layout_height="wrap_content"  android:layout_weight="1"/>

        <TextView
          android:text="第二行"  android:textSize="15pt"  android:gravity="center_vertical"
          android:background="#00aa00"  android:layout_width="fill_parent"
          android:layout_height="wrap_content"  android:layout_weight="1"/>

        <TextView
          android:text="第三行"  android:textSize="15pt"  android:gravity="center_vertical"
          android:background="#0000aa"  android:layout_width="fill_parent"
          android:layout_height="wrap_content"  android:layout_weight="1"/>

        <TextView
          android:text="第四行"  android:textSize="15pt"  android:gravity="center_vertical"
          android:background="#aaaa00"  android:layout_width="fill_parent"
          android:layout_height="wrap_content"  android:layout_weight="1"/>
</LinearLayout>
```

main.xml： horizontal

```
<Linearlayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="horizontal"
android:layout_width="fill_parent"  android:layout_height="fill_parent"  >
<TextView
          android:text="第一列"  android:gravity="center_horizontal"
          android:background="#aa0000"  android:layout_width="wrap_content"
          android:layout_height="fill_parent"  android:layout_weight="1"/>

        <TextView
          android:text="第二列"  android:gravity="center_horizontal"
          android:background="#00aa00"  android:layout_width="wrap_content"
          android:layout_height="fill_parent"  android:layout_weight="1"/>

        <TextView
          android:text="第三列"  android:gravity="center_horizontal"
          android:background="#0000aa"  android:layout_width="wrap_content"
          android:layout_height="fill_parent"  android:layout_weight="1"/>

        <TextView
          android:text="第四列"  android:gravity="center_horizontal"
          android:background="#aaaa00"  android:layout_width="wrap_content"
          android:layout_height="fill_parent"  android:layout_weight="1"/></Linearlayout>
```

3.3.2 AbsoluteLayout（绝对布局）

绝对布局根据设定好的坐标进行定位显示

AbsoluteLayout 两个重要的属性：
android:layout_x 组件在屏幕中的 X 坐标
android:layout_y 组件在屏幕中的 Y 坐标

3.3.3 RelativeLayout（相对布局）

是按照相对某个组件的位置来进行布局，也就是说参考某个组件，置于此组件的上、下、左、右

其中几个重要的属性：
android:layout_below=“组件 ID” 在某组件下面
android:layout_above=“组件 ID” 在某组件上面
android:layout_toRightOf=“ID” 在某组件右边
android:layout_toLeftOf=“ID” 在某组件左边

RelativeLayout Demo

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"  android:layout_height="fill_parent">
  <TextView android:id="@+id/label"
    android:layout_width="fill_parent"  android:layout_height="wrap_content"
    android:text="请输入:"/>
  <EditText  android:id="@+id/entry"
    android:layout_width="fill_parent"  android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"
    android:layout_below="@id/label"/>
  <Button     android:id="@+id/ok"
    android:layout_width="wrap_content"  android:layout_height="wrap_content"
    android:layout_below="@id/entry"   android:layout_alignParentRight="true"
    android:layout_marginLeft="10dip"  android:text="确定" />

  <Button  android:layout_width="wrap_content"
    android:layout_height="wrap_content"      android:layout_toLeftOf="@id/ok"
    android:layout_alignTop="@id/ok"      android:text="取消" />
</RelativeLayout>
```

3.3.4 TableLayout（表格布局）

是比较常用的布局，它是按照表格的方式来布局整个画面的

TableRow：TableLayout 中需要嵌入行，然后将组件置于 TableRow 中才能显示成 Table 的形式

几个重要的属性：
android:layout_weight：比重
TableRow：行

TableLayout Demo

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:stretchColumns="1">
  <TableRow>
    <TextView
      android:layout_column="1"        android:text="打开..."        android:padding="3dip" />
    <TextView
      android:text="Ctrl-O"        android:gravity="right"        android:padding="3dip" />
  </TableRow>
  <TableRow>
    <TextView
      android:layout_column="1"        android:text="保存..."        android:padding="3dip" />
    <TextView
      android:text="Ctrl-S"        android:gravity="right"        android:padding="3dip" />
  </TableRow>
  <TableRow>
    <TextView
      android:layout_column="1"        android:text="另存为..."        android:padding="3dip" />
    <TextView
      android:text="Ctrl-Shift-S"        android:gravity="right"        android:padding="3dip" />
  </TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />
  <TableRow>
    <TextView        android:text="*"        android:padding="3dip" />
    <TextView
      android:text="导入..."        android:padding="3dip" />
  </TableRow>
  <TableRow>
    <TextView        android:text="*"        android:padding="3dip" />
    <TextView        android:text="导出..."        android:padding="3dip" />
    <TextView        android:text="Ctrl-E"    android:gravity="right"    android:padding="3dip" />
  </TableRow>

  <View
    android:layout_height="2dip"    android:background="#FF909090" />

  <TableRow>
    <TextView
      android:layout_column="1"        android:text="退出"
      android:padding="3dip" />
  </TableRow>
</TableLayout>
```

3.3.5 FrameLayout（框架布局）

是一个比较特殊的布局

此布局一般放一个组件，并且这个组件是靠左上角显示，

如果加入多个组件，那将会显示最上层的一个组件。

FrameLayout Demo

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
        <ImageView
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:scaleType="center"
                android:src="@drawable/golden_gate"
                />
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginBottom="20dip"
                android:layout_gravity="center_horizontal|bottom"
                android:padding="12dip"
                android:background="#AA000000"
                android:textColor="#ffffffff"
                android:text="Golden Gate"
                />
</FrameLayout>
```

3.3.6 布局之间的关系

LinearLayout、AbsoluteLayout、RelativeLayout、FrameLayout 均是 ViewGroup 的子类

TableLayout 则是 LinearLayout 子类，如果 TableLayout 中的组件没有放入 TableRow 中的话，那么就会按照 LinearLayout 显示

在 Android 中，布局是可以相互嵌套的，比如 LinearLayout 中能够嵌入 TableLayout 一样

3.4 样式和主题(style&theme)

android中的样式和CSS样式作用相似，都是用于为界面元素定义显示风格，它是一个包含一个或者多个view控件属性的集合。如：需要定义字体的颜色和大小。

在CSS中是这样定义的：

```
<style>
   .lxt{COLOR:#0000CC;font-size:18px;}
</style>
```

可以像这样使用上面的css样式：`<div class="lxt">lxt008</div>`

在Android中可以这样定义样式：

在res/values/styles.xml文件中添加以下内容

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="lxt"> <!-- 为样式定义一个全局唯一的名字-->
      <item name="android:textSize">18px</item> <!-- name属性为样式要用在的View控件持有的属性-->
      <item name="android:textColor">#0000CC</item>
   </style>
</resources>
```

在layout文件中可以像下面这样使用上面的android样式：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemes.android.com/apk/res/android" ...>
   <TextView style="@style/lxt"
      ..... />
</LinearLayout>
```

<style>元素中有一个parent属性。这个属性可以让当前样式继承一个父样式，当前样式可以继承到父样式的值。当然，如果父样式的值不符合你的需求，你也可以对它进行修改，如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="lxt">
      <item name="android:textSize">18px</item> <!-- name属性为样式要用在的View控件持有的属性 -->
      <item name="android:textColor">#0000CC</item>
   </style>
   <style name="sublxt" parent="@style/lxt">
      <item name="android:textColor">#FF0000</item>
   </style>
</resources>
```

android中主题也是用于为应用定义显示风格，它的定义和样式的定义相同，如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="lxtTheme">
        <item name="android:windowNoTitle">true</item> <!-- 没标题 -->
        <item name="android:windowFullscreen">?android:windowNoTitle</item> <!-- 全屏显示 -->
</style>
</resources>
```

上面“?android:windowNoTitle”中的问号用于引用在当前主题中定义过的资源的值。

下面代码显示在AndroidManifest.xml中如何为应用设置上面定义的主题：

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
    android:theme="@style/lxtTheme">
    ......
</application>
```

除了可以在AndroidManifest.xml中设置主题，同样也可以在代码中设置主题，如下：

```
setTheme(R.style.lxtTheme);
```

尽管在定义上，样式和主题基本相同，但是它们使用的地方不同。样式用在单独的View，如：EditText、TextView等；主题通过AndroidManifest.xml中的<application>和<activity>用在整个应用或者某个Activity，主题对整个应用或某个Activity存在全局性影响。如果一个应用使用了主题，同时应用下的view也使用了样式，那么当主题与样式属性发生冲突时，样式的优先级高于主题。

另外android系统也定义了一些主题，例如：<activity android:theme="@android:style/Theme.Dialog">，该主题可以让Activity看起来像一个对话框。如果需要查阅这些主题，可以在文档的reference->android->R.style 中查看。

第四天.基础 UI 控件

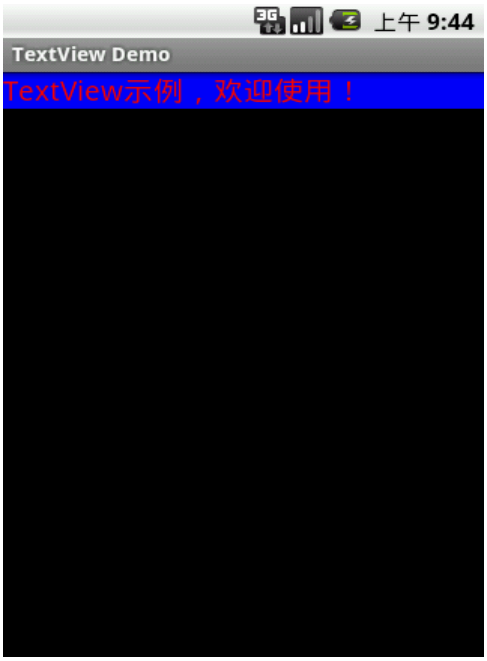
4.1 基本控件介绍

1、Button　　按钮

- 研究 **ButtonDemo**

2、TextView　　文本框

- 研究 **TextViewDemo**



3、EditText　　文本编辑框

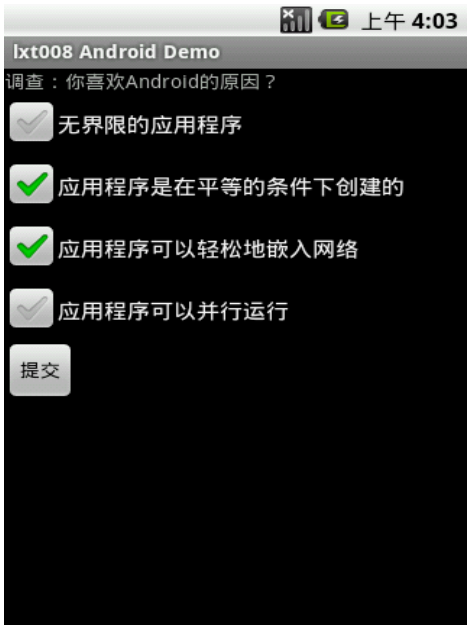
- **研究 EditTextDemo**

4、CheckBox　　多项选择

每个多选框都是独立的，可以通过迭代所有多选框，然后根据其状态是否被选中再获取其值。

- CheckBox.setChecked(true);//设置成选中状态。
- CheckBox.getText();//获取多选框的值
- 调用 setOnCheckedChangeListener() 方法，处理多选框被选择事件，把 CompoundButton.OnCheckedChangeListener 实例作为参数传入

- **研究 CheckBoxDemo**



5、RadioButton　　单项选择

- 研究 **RadioButtonDemo**

6、Spinner　　下拉列表

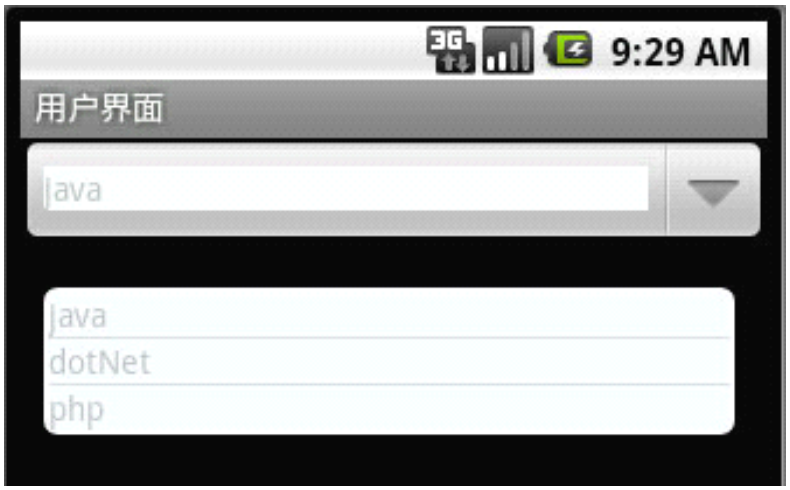
- Spinner.getItemAtPosition(Spinner.getSelectedItemPosition());获取下拉列表框的值
- 调用 setOnItemSelectedListener()方法，处理下拉列表框被选择事件，把 AdapterView.OnItemSelectedListener 实例作为参数传入

下拉列表框—Adapter

- 很多时候显示在下拉列表框的值并不是希望得到的值，如果要做一个联系人下拉列表框，列表框列出的是联系人的姓名，因为姓名有可能相同，所以我们希望得到的值应该为该联系人的 id，要实现这种需求我们需要自定义 Adapter，当然自定义 Adapter 需要我们编写一小段代码，如果我们不想编写 Adapter，又能实现我们的需求，那是最好不过的了。通过观察 ArrayAdapter 中 getView(int position, View convertView, ViewGroup parent)的内部代码发现，如果为 ArrayAdapter 指定的实际泛

型参数类型没有实现 CharSequence（字符串）接口，将会调用该类型对象的 toString()向下拉列表框输出显示值。利用这个特点我们可以重写 javaBean 的 toString()向下拉列表框提供显示值。

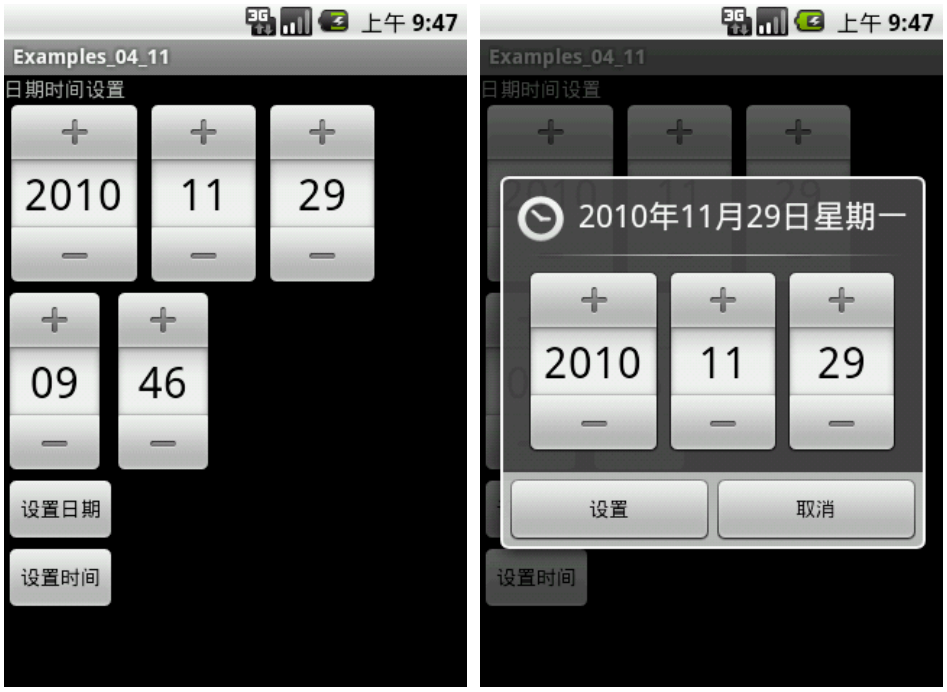
下拉列表框--自定义选项界面样式



- Spinner.getItemAtPosition(Spinner.getSelectedItemPosition());获取下拉列表框的值
- 调用 setOnItemSelectedListener()方法，处理下拉列表框被选择事件，把 AdapterView.OnItemSelectedListener 实例作为参数传入

7、TimePicker/DatePicker　时间选择器

- **研究 TimeDatePicker**



8、ScrollView　　滚动视图

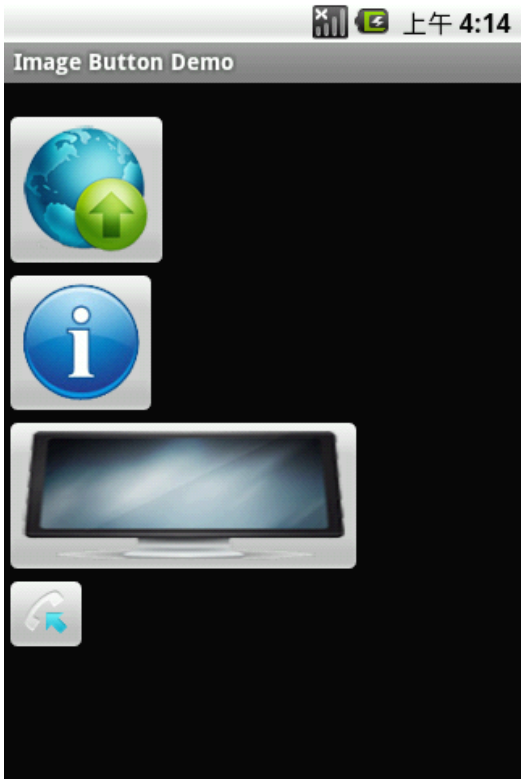
- **研究 ScrollViewDemo**

9、ImageView　　图片视图

- 研究 **ImageViewDemo**

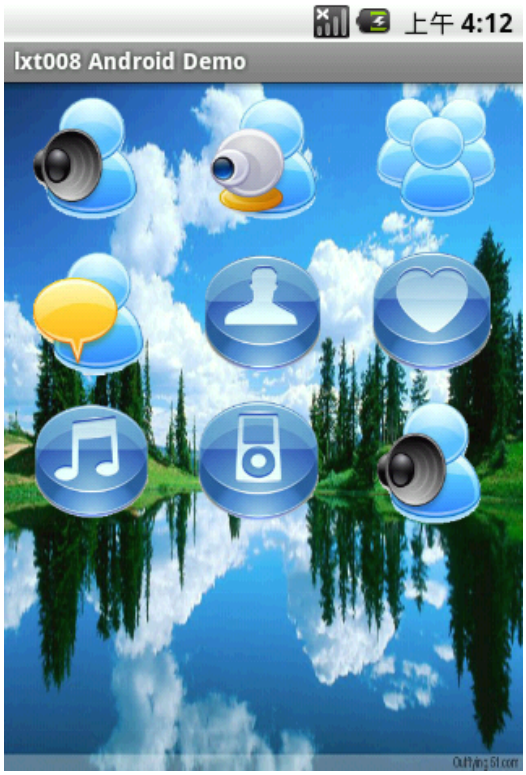
10、ImageButton　　图片按钮

- 研究 **ImageButtonDemo**



11、GridView　　网格视图

- **研究 GridViewImageDemo**



4.2 认识 Widget 组件

1、界面中展示的各个小组件

2、有独立的事件处理能力

3、所有 Wiget 组件都是继承 View 而来

4.3 Widget 组件类继承关系

第五天.高级 UI 控件

高级控件介绍

12、ProgressBar　　进度条

创建进度条

在布局 xml 文件中添加进度条代码:

```
<ProgressBar android:layout_width="fill_parent" android:layout_height="20px"
    style="?android:attr/progressBarStyleHorizontal"
    android:id="@+id/downloadbar"/>
```

在代码中操作进度条:

```
ProgressBar.setMax(100);//设置总长度为 100
ProgressBar.setProgress(0);//设置已经开启长度为 0,假设设置为 50,进度条将进行到一半
```

- 研究 **ProgressBarDemo**

13、ProgressDialog　　进度条对话框

```
打开对话框
new AlertDialog.Builder(this)
.setTitle(“lxt008")
.setMessage("浏览 http://www.lxt008.com")
      .setPositiveButton("打开链接",
      new DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialoginterface, int i){
      Uri uri = Uri.parse("http://www.lxt008.com/");
      Intent intent = new Intent(Intent.ACTION_VIEW, uri);
      startActivity(intent);
        }
      }
).show();
```

- 研究 **DialogDemo**

带进度条的对话框

- 研究 **ProgressDialogDemo**

进度对话框 ProgressDialog

- 使用代码 ProgressDialog.show(ProgressDialogActivity.this, "请稍等", "数据正在加载中...", true);创建并显示一个进度对话框。
- 调用 setProgressStyle()方法设置进度对话框风格。有两种风格：
  ProgressDialog.STYLE_SPINNER 旋体进度条风格 (为默认风格)
  ProgressDialog.STYLE_HORIZONTAL 横向进度条风格

14、SeekBar 拖动条

- **SeekBar.getProgress()获取拖动条当前值**
- **调用 setOnSeekBarChangeListener() 方法处理拖动条值变化事件，把 SeekBar.OnSeekBarChangeListener 实例作为参数传入**

➢ **研究 SeekBarDemo**

15、ImageSwitcher 图片切换

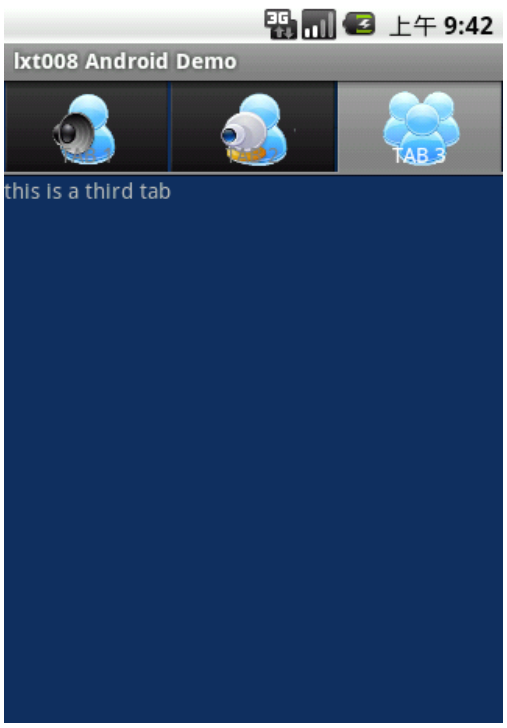
- 研究 **ImageSwitcherDemo**

16、Gallery 图片切换

- 研究 **GalleryDemo**

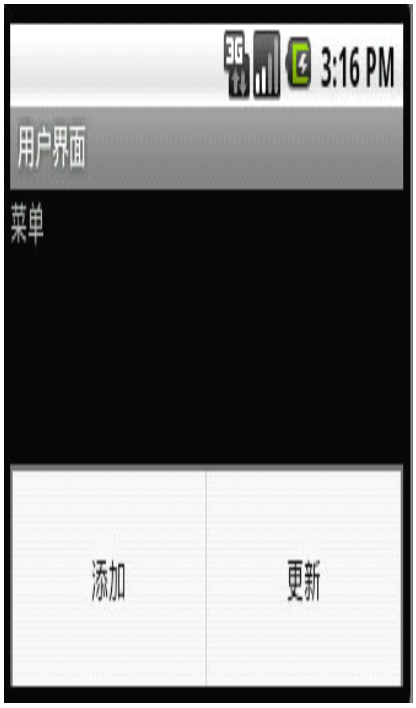
17、TabHost　　标签组件

- 研究 **TabHostDemo**



18、Menu　　菜单

- 重写 Activity 的 onCreateOptionsMenu(Menu menu)方法，该方法用于创建选项菜单，在用户按下手机的“Menu”按钮时就会显示创建好的菜单，在 onCreateOptionsMenu(Menu menu)方法内部可以调用 Menu.add()方法实现菜单的添加。
- 重写 Activity 的 onMenuItemSelected()方法，该方法用于处理菜单被选择事件



第六天.Android Service

6.1 Service 概述

6.1.1 Service 概念及用途

- 服务是运行在后台的一段代码。
- 不是进程，也不是线程。
- 可以运行在它自己的进程,也可以运行在其他应用程序进程的上下文（**context**）里面，这取决于自身的需要。
- **Android** 中的服务，它与 **Activity** 不同，它是不能与用户交互的，不能自己启动的，运行在后台的程序。
- 媒体播放器的服务，当用户退出媒体选择用户界面，仍然希望音乐依然可以继续播放，这就是由服务（**service**）来保证当用户界面关闭时音乐继续播放的。
- 比如当我们一个应用的数据是通过网络获取的，不同时间的数据是不同的,这时候我们可以用 **Service** 在后台定时更新，而不用每打开应用的时候在去获取。

6.2 Service 生命周期

- onCreate()
  - 在服务被创建时调用，该方法只会被调用一次，无论调用多少次 startService()或 bindService()方法，服务也只被创建一次。
- onStart()
  - 只有采用 Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。多次调用 startService()方法尽管不会多次创建服务，但 onStart() 方法会被多次调用。
- onDestroy()
  - 服务被终止时调用。
- onBind()
  - 只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，当调用者与服务已经绑定，多次调用 Context.bindService()方法并不会导致该方法被多次调用。
- onUnbind()
  - 只有采用 Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用。
- **startService** 后，即使调用 **startService** 的进程结束了 **Service** 仍然还存在，直到有进程调用 **stopService**，或者 **Service** 自己自杀（**stopSelf()**）。
- **bindService** 后，**Service** 就和调用 **bindService** 的进程同生共死了，也就是说当调用 **bindService** 的进程死了，那么它 **bind** 的 **Service** 也要跟着被结束，当然期间也可以调用 **unbindservice** 让 **Service** 结束。

- **两种方式混合使用时，比如说你 startService 了，我 bindService 了，那么只有你 stopService 了而且我也 unbindservice 了，这个 Service 才会被结束。**

6.3 启动与停止 Service

6.3.1 Service 开发步骤

第一步：继承 Service 类

```
public class MyService extends Service { }
```

第二步：在 AndroidManifest.xml 文件中的<application>节点里对服务进行配置:

```
<service android:name=".MyService" />
```

服务不能自己运行，使用 startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。使用 bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止，大有“不求同时生，必须同时死”的特点。

如果打算采用 Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的 onCreate()方法，接着调用 onStart()方法。如果调用 startService()方法前服务已经被创建，多次调用 startService()方法并不会导致多次创建服务，但会导致多次调用 onStart()方法。采用 startService()方法启动的服务，只能调用 Context.stopService()方法结束服务，服务结束时会调用 onDestroy()方法。

如果打算采用 Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的 onCreate()方法，接着调用 onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的 onUnbind()方法，接着调用 onDestroy()方法。如果调用 bindService()方法前服务已经被绑定，多次调用 bindService()方法并不会导致多次创建服务及绑定(也就是说 onCreate()和 onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务解除绑定，可以调用 unbindService()方法，调用该方法也会导致系统调用服务的 onUnbind()-->onDestroy()方法。

6.3.2 采用 startService()启动服务

采用Context.startService()方法启动服务的代码如下：

```
public class HelloActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        ......
        Button button =(Button) this.findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener(){
          public void onClick(View v) {
                    Intent intent = new Intent(HelloActivity.this, SMSService.class);
                    startService(intent);
          });
    }
}
```

6.3.3 采用 bindService()启动服务

采用Context.startService()方法启动服务的代码如下：

```
public class HelloActivity extends Activity {
    ServiceConnection conn = new ServiceConnection() {
          public void onServiceConnected(ComponentName name, IBinder service) {
          }
          public void onServiceDisconnected(ComponentName name) {
          }
    };
public void onCreate(Bundle savedInstanceState) {
       Button button =(Button) this.FindViewById(R.id.button);
       button.setOnClickListener(new View.OnClickListener(){
          public void onClick(View v) {
                    Intent intent = new Intent(HelloActivity.this, SMSService.class);
                    bindService(intent, conn, Context.BIND_AUTO_CREATE);
                    //unbindService(conn);//解除绑定
          });
    }
}
```

6.3.4 Service 服务演示

- **1.**新建一个 **Android** 工程 **ServiceDemo**
- **2.**修改 **main.xml** 代码，增加二个按钮
- **3.**新建一个 **Service**，命名为 **MyService.java**
- **4.**新建 **ServiceDemo.java**
- **5.**配置 **AndroidManifest.xml**
- **6.**执行上述工程，用 **Logcat** 查看日志
- **7.**按 **HOME** 键进入 **Settings(**设置**)→Applications(**应用）**→Running Services(**正在运行的服务）

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"    android:layout_width="fill_parent"
    android:layout_height="fill_parent"    >
  <TextView
    android:id="@+id/text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
  <Button
    android:id="@+id/startservice"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="startService" />
 <Button
    android:id="@+id/stopservice"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="stopService" />
</LinearLayout>
```

MyService.java

```
public class MyService extends Service {
  //定义个一个Tag标签
  private static final String TAG = "MyService";
  //一个Binder类，用在onBind()方法里，这样Activity那边可以获取到
  private MyBinder mBinder = new MyBinder();
  public IBinder onBind(Intent intent) {
    Log.e(TAG, "start IBinder~~~~");
    return mBinder;
  }
  public void onCreate() {
    Log.e(TAG, "start onCreate~~~~");
    super.onCreate();
  }
  public void onStart(Intent intent, int startId) {
    Log.e(TAG, "start onStart~~~~");
    super.onStart(intent, startId);
  }

 public void onDestroy() {
   Log.e(TAG, "start onDestroy~~~~");
   super.onDestroy();
 }
 public boolean onUnbind(Intent intent) {
   Log.e(TAG, "start onUnbind~~~~");
   return super.onUnbind(intent);
 }
 public String getSystemTime(){
   Time t = new Time();
   t.setToNow();
   return t.toString();
 }
 public class MyBinder extends Binder{
   MyService getService()
   {
     return MyService.this;
   }
 } }
```

ServiceDemo.java

```
public class ServiceDemo extends Activity implements OnClickListener {
    private MyService  mMyService;
    private TextView mTextView;
    private Button startServiceButton;
    private Button stopServiceButton;

    private Context mContext;

    public void onCreate(Bundle  savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        setupViews();
    }
```

```
    public void setupViews(){
        mContext = ServiceDemo.this;
        mTextView = (TextView)findViewById(R.id.text);

        startServiceButton = (Button)findViewById(R.id.startservice);
        stopServiceButton = (Button)findViewById(R.id.stopservice);

        startServiceButton.setOnClickListener(this);
        stopServiceButton.setOnClickListener(this);

    }
```

```
    public void onClick(View v) {
        if(v == startServiceButton){
            Intent i  = new Intent();
            i.setClass(ServiceDemo.this,  MyService.class);
            mContext.startService(i);
        }else if(v == stopServiceButton){
            Intent i  = new Intent();
            i.setClass(ServiceDemo.this,  MyService.class);
            mContext.stopService(i);
        }
    }
}
```

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lxt008"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"  android:label="@string/app_name">
        <activity android:name=".ServiceDemo"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".MyService" android:exported="true"></service>
    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

6.4 Notification 通知

6.4.1 Android 中的通知(Notification)

通知用于在状态栏显示消息，消息到来时以图标方式表示，如下：

如果需要查看消息，可以拖动状态栏到屏幕下方即可查看消息。

发送消息的代码如下：

```
//获取通知管理器
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
int icon = android.R.drawable.stat_notify_chat;
long when = System.currentTimeMillis();
//新建一个通知，指定其图标和标题
//第一个参数为图标,第二个参数为标题,第三个为通知时间
Notification notification = new Notification(icon, null, when);

Intent openintent = new Intent(this, OtherActivity.class);
//当点击消息时就会向系统发送openintent意图
PendingIntent contentIntent = PendingIntent.getActivity(this, 0, openintent, 0);
notification.setLatestEventInfo(this,"标题 ","我是内容", contentIntent);
mNotificationManager.notify(0,notification);
```

6.5 案例分析

- **参考案例：NotificationDemo**

第七天.SharedPreferences 与文件

7.1 SharedPreferences

7.1.1 数据存储方式

- **Shared Preferences**
- 存储到文件
- **SQLite** 数据库
- 存储到网络
- 内容提供者（**Content provider**）

7.1.2 SharedPreferences

- 如何保存软件配置参数：
  - **Window:** 采用 **ini** 文件进行
  - **j2se** 应用**:** 采用 **properties** 属性文件
  - **Android** 平台提供一个 **SharedPreferences** 类，它是一个轻量级的存储类，适合用于保存软件配置参数。
  - 使用 **SharedPreferences** 保存数据，其背后是用 **xml** 文件存放数据，文件存放在**/data/data/<package name>/shared_prefs** 目录下

7.1.3 SharedPreferences 存储数据

- **SharedPreferences sharedPreferences = getSharedPreferences("lxt008", Context.MODE_PRIVATE);**

Editor editor = sharedPreferences.edit();//获取编辑器
editor.putString("name", "lxt");
editor.putInt("age", 35);
editor.commit();//提交修改
生成的 **lxt008.xml** 文件内容如下：
<?xml version=“1.0” encoding=“utf-8” standalone=“yes” ?>
<map>
<string name="name">lxt</string>
<int name="age" value=“30" />
</map>

- **getSharedPreferences(name,mode)**方法
  - 参数 **1:**指定该文件名称，名称不用带后缀。
  - 参数 **2:**指定文件的操作模式，共有四种操作模式。
    - ✓ **Context.MODE_PRIVATE**：为默认操作模式，代表该文件是私有数据，只能被应用本身访问，在该模式下，写入的内容会覆盖原文件的内容，如果想把新写入的内容追加到原文件中。可以使用 **Context.MODE_APPEND**
    - ✓ **Context.MODE_APPEND**：模式会检查文件是否存在，存在就往文件追加内容，否则就创建新文件。
    - ✓ **Context.MODE_WORLD_READABLE** 和 **Context.MODE_WORLD_WRITEABLE** 用来控制其他应用是否有权限读写该文件。
    - ✓ **MODE_WORLD_READABLE**：表示当前文件可以被其他应用读取；**MODE_WORLD_WRITEABLE**：表示当前文件可以被其他应用写入。
- **getPreferences(mode)**方法操作 **SharedPreferences**，这个方法默认使用当前类不带包名的类名作为文件的名称。
- **android** 有一套自己的安全模型，当应用程序**(.apk)**在安装时系统就会分配给他一个 **userid**，当该应用要去访问其他资源比如文件的时候，就需要 **userid** 匹配。
- 默认情况下，应用创建的文件**/sharedpreferences/**数据库都应该是私有的（位于**/data/data/**包名），其他程序无法访问。
- 除非在创建时指定了 **Context.MODE_WORLD_READABLE** 或者 **Context.MODE_WORLD_WRITEABLE** ，只有这样其他程序才能正确访问。

7.1.4 访问 SharedPreferences 数据

- 访问 **SharedPreferences** 中的数据代码如下：

```
SharedPreferences sharedPreferences = getSharedPreferences("lxt008", Context.MODE_PRIVATE);
//getString()第二个参数为缺省值，如果 preference 中不存在该 key，将返回缺省值
String name = sharedPreferences.getString("name", "");
int age = sharedPreferences.getInt("age", 1);
```

7.2 不同应用共享数据

7.2.1 访问其他应用 SharedPreferences 数据

- 访问其他应用 **Preference**，前提条件是：该 **preference** 创建时指定了 **Context.MODE_WORLD_READABLE** 或者 **Context.MODE_WORLD_WRITEABLE**。
  - **getSharedPreferences(“lxt008", Context.MODE_WORLD_READABLE);**
  - 其他应用要访问上面应用的 **preference**，首先需要创建上面应用的 **Context**，然后通过 **Context** 访问 **preference,**访问 **preference** 时会在应用所在包下的 **shared_prefs** 目录找到 **preference** ：

```
Context otherAppsContext = createPackageContext( “ com.lxt008", Context.CONTEXT_IGNORE_SECURITY);
SharedPreferences sharedPreferences = otherAppsContext.getSharedPreferences( “lxt008", Context.MODE_WORLD_READABLE);
String name = sharedPreferences.getString("name", "");
int age = sharedPreferences.getInt("age", 0);
```

Shared Preferences 案例

- 参考：**SharedPreferencesMidiPlayer**

结果验证

- 切换到 **DDMS** 视图
- 选择 **File Explorer** 标签
- **/data/data/**你的项目目录**/shared_prefs**

7.3 Android 文件操作

7.3.1 文件存储

- 文件可以存储比使用引用更大数量的数据
- Android 提供方法来读、写文件
- 只有本地文件可以被访问
- 优点：可以存储大容量的数据
- 缺点：文件更新或是格式改变可能会导致巨大的编程工作

7.3.2 读文件操作

- **Context.openFileInput(String name)**
- 打开一个与应用程序联系的私有文件输入流
- 当文件不存在时抛出 **FileNotFoundException**
- **FileInputStream in = this.openFileInput("test.txt");//**打开文件**"test.txt"**
- **……**
- **in.close();//**关闭输入流

7.3.3 写文件操作

- **Context.openFileOutput(String name,int mode)**
  - 开启一个与应用程序联系的私有文件输出流
- 当文件不存在时该文件将被创建
- 文件输出流可以在添加模式中打开，这意味新的数据将被添加到文件的末尾
- **FileOutputStream out = this.openFileOutput("test2.txt",MODE_APPEND);**
- **//**打开文件**"test2.txt"**进行写操作、使用 **MODE_APPEND** 在添加模式中打开文件
- **……**
- **out.close();//**关闭输出流

7.3.4 读取静态文件

- 要打开打包在程序中的静态文件，使用 **Resources.openRawResource(R.raw.mydatafile)**
- 该文件必须放在文件夹 **res/raw/**中
- **InputStreamin = this.getResources().openRawResource(R.raw.my);**
- **…** **//**获得 **Context** 资源
- **in.close();//**关闭输入流

文件操作案例

- 参考：**FileMidiPlayer**

7.3.5 使用文件进行数据存储

首先给大家介绍使用文件如何对数据进行存储，Activity 提供了 openFileOutput()方法可以用于把数据输出到文件中，具体的实现过程与在 J2SE 环境中保存数据到文件中是一样的。

```
public class FileActivity extends Activity {
    @Override public void onCreate(Bundle savedInstanceState) {
        ...
        FileOutputStream outStream = this.openFileOutput("lxt008.txt",
Context.MODE_PRIVATE);
        outStream.write("lxt008".getBytes());
        outStream.close();
    }
}
```

openFileOutput()方法的第一参数用于指定文件名称，不能包含路径分隔符“/”，如果文件不存在，Android 会自动创建它。创建的文件保存在/data/data/<package name>/files 目录，如：/data/data/com.lxt008/files/lxt008.txt，通过点击 Eclipse 菜单“Window”-“Show View”-“Other”，在对话窗口中展开 android 文件夹，选择下面的 File Explorer 视图，然后在 File Explorer 视图中展开/data/data/<package name>/files 目录就可以看到该文件。

openFileOutput()方法的第二参数用于指定操作模式，有四种模式，分别为：

Context.MODE_PRIVATE = 0
Context.MODE_APPEND = 32768
Context.MODE_WORLD_READABLE = 1
Context.MODE_WORLD_WRITEABLE = 2

7.3.6 读取文件内容

如果要打开存放在/data/data/<package name>/files 目录应用私有的文件，可以使用 Activity 提供 openFileInput()方法。

```
FileInputStream inStream = this.getContext().openFileInput("lxt008.txt");
Log.i("FileTest", readInStream(inStream));
```

readInStream()的方法请看本页下面备注。

或者直接使用文件的绝对路径：

```
File file = new File("/data/data/com.lxt008/files/lxt008.txt");
FileInputStream inStream = new FileInputStream(file);
Log.i("FileTest", readInStream(inStream));
```

注意：上面文件路径中的“com.lxt008”为应用所在包，应替换为你自己应用使用的包。

对于私有文件只能被创建该文件的应用访问，如果希望文件能被其他应用读和写，可以在创建文件时，指定 Context.MODE_WORLD_READABLE 和 Context.MODE_WORLD_WRITEABLE 权限。
Activity 还提供了 getCacheDir()和 getFilesDir()方法：
getCacheDir()方法用于获取/data/data/<package name>/cache 目录
getFilesDir()方法用于获取/data/data/<package name>/files 目录

7.4 SDCard 文件存取

7.4.1 把文件存放在 SDCard

使用 Activity 的 openFileOutput()方法保存文件，文件是存放在手机空间上，一般手机的存储空间不是很大，存放些小文件还行，如果要存放像视频这样的大文件，是不可行的。对于像视频这样的大文件，我们可以把它存放在 SDCard。 SDCard 是干什么的？你可以把它看作是移动硬盘或 U 盘。
在模拟器中使用 SDCard，你需要先创建一张 SDCard 卡（当然不是真的 SDCard，只是镜像文件）。创建 SDCard 可以在 Eclipse 创建模拟器时随同创建，也可以使用 DOS 命令进行创建，如下：
在 Dos 窗口中进入 android SDK 安装路径的 tools 目录，输入以下命令创建一张容量为 2G 的 SDCard，文件后缀可以随便取，建议使用.img：
mksdcard 2048M D:\AndroidTool\sdcard.img
在程序中访问 SDCard，你需要申请访问 SDCard 的权限。
在 AndroidManifest.xml 中加入访问 SDCard 的权限如下:

```
<!-- 在 SDCard 中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 往 SDCard 写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

要往 SDCard 存放文件，程序必须先判断手机是否装有 SDCard，并且可以进行读写。
注意：访问 SDCard 必须在 AndroidManifest.xml 中加入访问 SDCard 的权限

```
if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
        File sdCardDir = Environment.getExternalStorageDirectory();//获取 SDCard 目录
        File saveFile = new File(sdCardDir, “lxt008.txt”);
FileOutputStream outStream = new FileOutputStream(saveFile);
outStream.write(“lxt008".getBytes());
outStream.close();
}
```

Environment.getExternalStorageState()方法用于获取 SDCard 的状态，如果手机装有 SDCard，并且可以进行读写，那么方法返回的状态等于 Environment.MEDIA_MOUNTED。
Environment.getExternalStorageDirectory()方法用于获取 SDCard 的目录，当然要获取 SDCard 的目录，你也可以这样写：

```
File sdCardDir = new File("/sdcard"); //获取 SDCard 目录
File saveFile = new File(sdCardDir, “lxt008.txt");
```

```
//上面两句代码可以合成一句：  File saveFile = new File("/sdcard/lxt008.txt");
FileOutputStream outStream = new FileOutputStream(saveFile);
outStream.write("lxt008".getBytes());
outStream.close();
```

使用Activity的openFileOutput()方法保存文件，文件是存放在手机空间上，一般手机的存储空间不是很大，存放些小文件还行，如果要存放像视频这样的大文件，是不可行的。对于像视频这样的大文件，我们可以把它存放在SDCard。SDCard是干什么的？你可以把它看作是移动硬盘或U盘。

在模拟器中使用SDCard，你需要先创建一张SDCard卡（当然不是真的SDCard，只是镜像文件）。创建SDCard可以在Eclipse创建模拟器时随同创建，也可以使用DOS命令进行创建，如下：

在Dos窗口中进入android SDK安装路径的tools目录，输入以下命令创建一张容量为2G的SDCard，文件后缀可以随便取，建议使用.img

```
mksdcard 2048M D:\AndroidTool\sdcard.img
```

在程序中访问SDCard，你需要申请访问SDCard的权限。

在AndroidManifest.xml中加入访问SDCard的权限如下：

```
<!-- 在SDCard中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 往SDCard写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

要往SDCard存放文件，程序必须先判断手机是否装有SDCard，并且可以进行读写。

注意：访问SDCard必须在AndroidManifest.xml中加入访问SDCard的权限

```
if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
        File sdCardDir = Environment.getExternalStorageDirectory();//获取SDCard目录
        File saveFile = new File(sdCardDir, "lxt008.txt");
        FileOutputStream outStream = new FileOutputStream(saveFile);
        outStream.write("lxt008".getBytes());
        outStream.close();
}
```

Environment.getExternalStorageState()方法用于获取SDCard的状态，如果手机装有SDCard，并且可以进行读写，那么方法返回的状态等于Environment.MEDIA_MOUNTED。

Environment.getExternalStorageDirectory()方法用于获取SDCard的目录，当然要获取SDCard的目录，你也可以这样写：

```
File sdCardDir = new File("/sdcard");//获取SDCard目录
File saveFile = new File(sdCardDir, "lxt008.txt");
//上面两句代码可以合成一句：  File saveFile = new File("/sdcard/lxt008.txt");
FileOutputStream outStream = new FileOutputStream(saveFile);
outStream.write("lxt008".getBytes());
outStream.close();
```

第八天.SQLite 数据库技术

8.1 SQLite 介绍

8.1.1 数据库存储

- 在某些情况下，文件不是有效的
  - 多线程数据访问
  - 需要事务处理
  - 如果应用程序处理可能变化的复杂数据结构
  - 数据库对于创建它们的包套件是私有的

8.1.2 SQLite 介绍

- **SQLite** 是一个轻量级的数据库，体积大小只用几千字节
- 一些 **SQL** 的指令只是部分支持，例如：**ALTER**、**TABLE**
- 广泛应用在嵌入式移动设备之上。
- 参阅 **http://www.sqlite.org** 获取更多信息

8.2 创建/打开/删除数据库

8.2.1 创建数据库

- **Context.createDatabase(**

String name, //
int version, //
int mode, //
CursorFactory factory //
)

- 创建一个新的数据库并返回一个 **SQLiteDatabase** 对象
- 数据库不能被创建，则抛出 **FileNotFoundException** 异常

8.2.2 其他创建数据库的方法

```
SQLiteDatabase mydataBase=SQLiteDatabase.create(new CursorFactory(){
//创建一个数据库
//工厂类，一个可选工厂类，当查询时调用来实例化一个光标
public Cursor newCursor(SQLiteDatabase db,
SQLiteCursorDriver masterQuery, String editTable,
SQLiteQuery query) {
return null;
}
});
SQLiteDatabase myDataBase=this.openOrCreateDatabase("myDataBase.db",
MODE_PRIVATE, new CursorFactory(){
//创建新的数据库，名称 myDatabase，模式 MODE_PRIVATE，游标工厂
//工厂类，一个可选工厂类，当查询时调用来实例化一个光标
public Cursor newCursor(SQLiteDatabase db,
SQLiteCursorDriver masterQuery, String editTable,
SQLiteQuery query) {
return null;
}
});
```

8.2.3 删除数据库

- **Context.deleteDatabase(String name)**
  - 删除指定名称的数据库
  - 假如数据库成功删除则返回 **true**，失败则为 **false**

8.2.4 打开数据库

- **Context.openDatabase(String file,CursorFactory factory)**
  - 打开一个存在的数据库并返回一个 **SQLiteDatabase** 对象
  - 如果数据库不存在则抛出 **FileNotFoundException** 异常
  - 如创建一个名为：**myDataBase** 的数据库，后缀为**.db**
  - **SQLiteDatabase my_DataBase=**

```
this.openOrCreateDatabase(
    "myDateBase.db",
    MODE_PRIVATE, null);
```

  - **my_DataBase.close();//**不要忘记关闭数据库

8.2.5 非查询 SQL 指令

- **SQLiteDatabase.execSQL(String sql)**
  - 可以用来执行非查询 **SQL** 指令，这些指令没有结果， 包括：**CREATE TABLE / DROP TABLE / INSERT** 等等。

8.3 创建/删除表

8.3.1 SQLite 基础案例

```
private SQLiteDatabase mSQLiteDatabase=null;
// 打开已经存在的数据库
mSQLiteDatabase = this.openOrCreateDatabase(DATABASE_NAME, MODE_PRIVATE,
null);
/* 在数据库 mSQLiteDatabase 中创建一个表 */
mSQLiteDatabase.execSQL(CREATE_TABLE);
/* 删除数据库 */
this.deleteDatabase(DATABASE_NAME);
/* 退出时，不要忘记关闭 */
mSQLiteDatabase.close();
/* 删除一个表 */
mSQLiteDatabase.execSQL("DROP TABLE " + TABLE_NAME);
/* 更新一条数据 */
ContentValues cv = new ContentValues();
cv.put(TABLE_NUM, miCount);
cv.put(TABLE_DATA, "修改后的数据" + miCount);
mSQLiteDatabase.update(TABLE_NAME, cv, TABLE_NUM + "=" +
Integer.toString(miCount - 1), null);
```

```
UpdataAdapter(); //更新界面


/*  向表中添加一条数据  */
      ContentValues cv = new ContentValues();
      cv.put(TABLE_NUM, miCount);
      cv.put(TABLE_DATA, "测试数据库数据" + miCount);
      mSQLiteDatabase.insert(TABLE_NAME, null, cv);
      miCount++;
      UpdataAdapter(); //更新界面

      /*  从表中删除指定的一条数据  */
      mSQLiteDatabase.execSQL("DELETE FROM " + TABLE_NAME + " WHERE _id="
+ Integer.toString(miCount));
      miCount--;
      if (miCount < 0){
            miCount = 0;
      }
      UpdataAdapter();   //更新界面
```

8.3.2 SQLite 基础案例：更新视图显示

```
/*  更新视图显示  */
public void UpdataAdapter(){
          //  获取数据库 Phones 的 Cursor
          Cursor cur = mSQLiteDatabase.query(TABLE_NAME, new String[] { TABLE_ID,
TABLE_NUM, TABLE_DATA }, null, null, null, null, null);
          miCount = cur.getCount();
          if (cur != null && cur.getCount() >= 0) {
               // ListAdapter 是 ListView 和后台数据的桥梁
               ListAdapter adapter = new SimpleCursorAdapter(this,
               //  定义 List 中每一行的显示模板
                    //  表示每一行包含两个数据项
                    android.R.layout.simple_list_item_2,
                    //  数据库的 Cursor 对象
                    cur,
                    //  从数据库的 TABLE_NUM 和 TABLE_DATA 两列中取数据
                    new String[] { TABLE_NUM, TABLE_DATA },
                    //  与 NAME 和 NUMBER 对应的 Views
                    new int[] { android.R.id.text1, android.R.id.text2 });
               /*  将 adapter 添加到 m_ListView 中  */
```

```
            m_ListView.setAdapter(adapter);
        }}
```

** 研究案例 **DatabaseDemo1**

8.4 CRUD 操作

8.4.1 查询 SQL 指令-游标 Cursors

- **Android** 使用游标(Cursors)来导航浏览查询结果
- 游标(Cursors)被 **android.database.Cursor** 对象来描述
- 一个游标(Cursors)是一个简单的指针，它从查询结果的一个元组跳到下一个元组(或前一个或第一个……)
- 游标(Cursors)在它定位位置的那一刻返回元组数据

```
//为创建 Cursor(游标)，必须执行查询，要么通过 SQL 使用 rawQuery()方法
//或是更精心设计的方法，像 query()方法
Cursor cur=my_DataBase.rawQuery("SELECT * FORM test", null);
if(cur!=null){//游标不为空
//返回给定名称的列的基于 0 开始的 index，如果该属性列不存在则返回-1
//通过它们的 index 来检索属性值
int numColumn=cur.getColumnIndex("someNumber");
if(cur.moveToFirst()){
//cur.moveToFirst()让游标指向第一行，如果游标指向第一行，则返回 true
do {
int num=cur.getInt(numColumn);//获得当前行该属性的值
/*Cursor 提供了不同的方法来回索不同的数据类型
例如 getInt(int index)/getString(int index)等等*/
/*做一些事情*/
} while (cur.moveToNext());
/*游标移动到下一行，如果游标已经通过了结果集中的最后，
即没有行可以移动时，则返回 false*/
//其他可能移动的是 previous() 和 first()方法
}
}
```

5.5 事务处理

5.5.1 使用事务操作 SQLite 数据库

使用 SQLiteDatabase 的 beginTransaction() 方法可以开启一个事务，程序执行到 endTransaction() 方法时会检查事务的标志是否为成功，如果为成功则提交事务，否则回滚

事务。当应用需要提交事务，必须在程序执行到 endTransaction()方法之前使用 setTransactionSuccessful() 方法设置事务的标志为成功，如果不调用 setTransactionSuccessful()方法，默认会回滚事务。使用例子如下：

```
 SQLiteDatabase db = ....;
db.beginTransaction();//开始事务
try {
      db.execSQL("insert into person(name, age) values(?,?)", new Object[]{“lxt008", 4});
      db.execSQL("update person set name=? where personid=?", new Object[]{“lxt008", 1});
      db.setTransactionSuccessful();//调用此方法会在执行到 endTransaction() 时提交当前事务，如果不调用此方法会回滚事务
} finally {
      db.endTransaction();//由事务的标志决定是提交事务，还是回滚事务
}
db.close();
```

上面两条 SQL 语句在同一个事务中执行。

其他

- **数据库辅助类**
  - **研究案例 DatabaseDemo2**
  - **数据库小工具**
  - **Sqlitebrowser 可以简单管理 Sqlite 数据库**

第九天.ContentProvider 与 BroadcastReceiver

9.1 ContentProvider

9.1.1 使用 ContentProvider 共享数据

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，如：采用文件方式对外共享数据，需要进行文件操作读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的好处是统一了数据访问方式。

当应用需要通过 ContentProvider 对外共享数据时，第一步需要继承 ContentProvider 并重写下面方法：

```
public class PersonContentProvider extends ContentProvider{
    public boolean onCreate()
    public Uri insert(Uri uri, ContentValues values)
```

public int delete(Uri uri, String selection, String[] selectionArgs)
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
public String getType(Uri uri)}

第二步需要在 AndroidManifest.xml 使用<provider>对该 ContentProvider 进行配置，为了能让其他应用找到该 ContentProvider ， ContentProvider 采用了 authorities（主机名/域名）对它进行唯一标识，你可以把 ContentProvider 看作是一个网站（想想，网站也是提供数据者），authorities 就是他的域名：

```
<manifest .... >
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <provider android:name=".PersonContentProvider"
android:authorities="com.lxt008.provider.personprovider"/>
    </application>
</manifest>
```

注意：一旦应用继承了 ContentProvider 类，后面我们就会把这个应用称为 ContentProvider（内容提供者）。

9.1.2 Uri 介绍

Uri 代表了要操作的数据，Uri 主要包含了两部分信息：1》需要操作的 ContentProvider ，2》对 ContentProvider 中的什么数据进行操作，一个 Uri 由以下几部分组成：

ContentProvider（内容提供者）的 scheme 已经由 Android 所规定， scheme 为：content://
主机名（或叫 Authority）用于唯一标识这个 ContentProvider，外部调用者可以根据这个标识来找到它。
路径（path）可以用来表示我们要操作的数据，路径的构建应根据业务而定，如下:
要操作 person 表中 id 为 10 的记录，可以构建这样的路径:/person/10
要操作 person 表中 id 为 10 的记录的 name 字段， person/10/name
要操作 person 表中的所有记录，可以构建这样的路径:/person
要操作 xxx 表中的记录，可以构建这样的路径:/xxx
当然要操作的数据不一定来自数据库，也可以是文件等他存储方式，如下:
要操作 xml 文件中 person 节点下的 name 节点，可以构建这样的路径：/person/name
如果要把一个字符串转换成 Uri，可以使用 Uri 类中的 parse()方法，如下：

```
Uri uri = Uri.parse("content://com.lxt008.provider.personprovider/person")
```

9.1.3 UriMatcher 类使用介绍

因为 Uri 代表了要操作的数据，所以我们很经常需要解析 Uri，并从 Uri 中获取数据。Android 系统提供了两个用于操作 Uri 的工具类，分别为 UriMatcher 和 ContentUris 。掌握它们的使用，会便于我们的开发工作。

UriMatcher 类用于匹配 Uri，它的用法如下：

首先第一步把你需要匹配 Uri 路径全部给注册上，如下：

```
//常量 UriMatcher.NO_MATCH 表示不匹配任何路径的返回码
UriMatcher   sMatcher = new UriMatcher(UriMatcher.NO_MATCH);
//如果 match()方法匹配 content://com.lxt008.provider.personprovider/person 路径，返回匹配码为 1
sMatcher.addURI(“cn.itcast.provider.personprovider”, “person”, 1);//添加需要匹配 uri，如果匹配就会返回匹配码
//如果 match()方法匹配 content://com.lxt008.provider.personprovider/person/230 路径，返回匹配码为 2
sMatcher.addURI(“com.lxt008.provider.personprovider”, “person/#”, 2);//#号为通配符
switch (sMatcher.match(Uri.parse("content://com.lxt008.provider.personprovider/person/10"))) {
    case 1
      break;
    case 2
      break;
    default://不匹配
      break;
}
```

注册完需要匹配的 Uri 后，就可以使用 sMatcher.match(uri)方法对输入的 Uri 进行匹配，如果匹配就返回匹配码，匹配码是调用 addURI()方法传入的第三个参数，假设匹配 content://com.lxt008.provider.personprovider/person 路径，返回的匹配码为 1

ContentUris 类用于获取 Uri 路径后面的 ID 部分，它有两个比较实用的方法：

withAppendedId(uri, id)用于为路径加上 ID 部分：

```
Uri uri = Uri.parse("content://com.lxt008.provider.personprovider/person")
Uri resultUri = ContentUris.withAppendedId(uri, 10);
//生成后的 Uri 为：content://com.lxt008.provider.personprovider/person/10
```

parseId(uri)方法用于从路径中获取 ID 部分：

```
Uri uri = Uri.parse("content://com.lxt008.provider.personprovider/person/10")
long personid = ContentUris.parseId(uri);//获取的结果为:10
```

9.1.4 使用 ContentProvider 共享数据

ContentProvider 类主要方法的作用：

public boolean onCreate()

该方法在 ContentProvider 创建后就会被调用， Android 在系统启动时就会创建 ContentProvider 。

public Uri insert(Uri uri, ContentValues values)

该方法用于供外部应用往 ContentProvider 添加数据。

public int delete(Uri uri, String selection, String[] selectionArgs)

该方法用于供外部应用从 ContentProvider 删除数据。

public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)

该方法用于供外部应用更新 ContentProvider 中的数据。

public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

该方法用于供外部应用从 ContentProvider 中获取数据。

public String getType(Uri uri)

该方法用于返回当前 Url 所代表数据的 MIME 类型。如果操作的数据属于集合类型，那么 MIME 类型字符串应该以 vnd.android.cursor.dir/开头，例如：要得到所有 person 记录的 Uri 为 content://com.lxt008.provider.personprovider/person，那么返回的 MIME 类型字符串应该为：“vnd.android.cursor.dir/person”。如果要操作的数据属于单一数据，那么 MIME 类型字符串应该以 vnd.android.cursor.item/开头，例如：得到 id 为 10 的 person 记录，Uri 为 content://com.lxt008.provider.personprovider/person/10，那么返回的 MIME 类型字符串应该为：“vnd.android.cursor.item/person”。

9.2 ContentResolver

9.2.1 ContentResolver

当外部应用需要对 ContentProvider 中的数据进行添加、删除、修改和查询操作时，可以使用 ContentResolver 类来完成，要获取 ContentResolver 对象，可以使用 Activity 提供的 getContentResolver()方法。 ContentResolver 类提供了与 ContentProvider 类相同签名的四个方法：

public Uri insert(Uri uri, ContentValues values)

该方法用于往 ContentProvider 添加数据。

public int delete(Uri uri, String selection, String[] selectionArgs)

该方法用于从 ContentProvider 删除数据。

public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)

该方法用于更新 ContentProvider 中的数据。

public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

该方法用于从 ContentProvider 中获取数据。

这些方法的第一个参数为 Uri，代表要操作的是哪个 ContentProvider 和对其中的什么数据进行操作，假设给定的是： Uri.parse(“content://com.lxt008.provider.personprovider/person/10”)，那么将会对主机名为 com.lxt008.provider.personprovider 的 ContentProvider 进行操作，操作的数据为 person 表中 id 为 10 的记录。

使用 ContentResolver 对 ContentProvider 中的数据进行添加、删除、修改和查询操作：

```
ContentResolver resolver =    getContentResolver();
Uri uri = Uri.parse(“content://com.lxt008.provider.personprovider/person");
//添加一条记录
ContentValues values = new ContentValues();
values.put("name", “lxt008");
values.put("age", 35);
resolver.insert(uri, values);
//获取 person 表中所有记录
Cursor cursor = resolver.query(uri, null, null, null, "personid desc");
while(cursor.moveToNext()){
      Log.i("ContentTest", "personid="+ cursor.getInt(0)+ ",name="+ cursor.getString(1));
}
//把 id 为 1 的记录的 name 字段值更改新为 liming
ContentValues updateValues = new ContentValues();
updateValues.put("name", "liming");
Uri updateIdUri = ContentUris.withAppendedId(uri, 2);
resolver.update(updateIdUri, updateValues, null, null);
//删除 id 为 2 的记录
Uri deleteIdUri = ContentUris.withAppendedId(uri, 2);
resolver.delete(deleteIdUri, null, null);
```

9.2.2 读取电话本

Demo 请参考 systemcontacts

进程间交互可以通过 ContentResolver 和 ContentProvider 类处理。

9.3 BroadcastReceiver

9.3.1 Broadcast Intent Receiver

当你想要写一个程序来对外部的事件做些处理时，可以使用 Broadcast Intent Receiver。比如:当电话响时，有短信时。Broadcast Intent Receiver 它并不能拿来显示 UI 画面，它必需利用 NotificationManager 来通知使用者他们感兴趣的事件发生了。

Broadcast Intent Receiver 同样的可以在 AndroidManifest.xml 中声明，但你也可以用写 Context.registerReceiver()程序的方式来注册你自己的 Broadcast Intent Receiver。你自己的程序并不会因为 BroadcastReceivers 被呼叫而被它执行起来。而是当 BroadcastReceiver 被触发时系统会依需求来执行相对应的程序。

程序可以利用 Context.sendBroadcast()来发出他们自己的 intent broadcast 给其它的程序。

9.3.2 广播接收者--BroadcastReceiver

广播接收者（BroadcastReceiver）用于异步接收广播 Intent，广播 Intent 的发送是通过调用 Context.sendBroadcast()、Context.sendOrderedBroadcast()或者 Context.sendStickyBroadcast()来实现的。通常一个广播 Intent 可以被订阅了此 Intent 的多个广播接收者所接收，广播接收者和 JMS 中的 Topic 消息接收者很相似。要实现一个广播接收者方法如下：

第一步：继承 BroadcastReceiver，并重写 onReceive()方法。

```
public class IncomingSMSReceiver extends BroadcastReceiver {
    @Override public void onReceive(Context context, Intent intent) {
    }
}
```

第二步：订阅感兴趣的广播 Intent，订阅方法有两种：

第一种：使用代码进行订阅

```
IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
IncomingSMSReceiver receiver = new IncomingSMSReceiver();
registerReceiver(receiver, filter);
```

第二种：在 AndroidManifest.xml 文件中的<application>节点里进行订阅:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

9.3.3 使用广播接收者窃听短信

如果你想窃听别人接收到的短信，达到你不可告人的目的，那么本节内容可以实现你的需求。当系统收到短信时，会发出一个 action 名称为 android.provider.Telephony.SMS_RECEIVED 的广播 Intent，该 Intent 存放了接收到的短信内容，使用名称“pdus”即可从 Intent 中获取短信内容。

```
public class IncomingSMSReceiver extends BroadcastReceiver {
private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
@Override public void onReceive(Context context, Intent intent) {
if (intent.getAction().equals(SMS_RECEIVED)) {
    SmsManager sms = SmsManager.getDefault();
    Bundle bundle = intent.getExtras();
    if (bundle != null) {
    Object[] pdus = (Object[]) bundle.get("pdus");
    SmsMessage[] messages = new SmsMessage[pdus.length];
    for (int i = 0; i < pdus.length; i++) messages[i] = SmsMessage.createFromPdu((byte[])
pdus[i]);
        for (SmsMessage message : messages){
            String msg = message.getMessageBody();
            String to = message.getOriginatingAddress();
            sms.sendTextMessage(to, null, msg, null, null);
}}}}}
```

在 AndroidManifest.xml 文件中的<application>节点里对接收到短信的广播 Intent 进行订阅:

```
<receiver android:name=".IncomingSMSReceiver">
<intent-filter><action
android:name="android.provider.Telephony.SMS_RECEIVED"/></intent-filter></receiver>
```

在 AndroidManifest.xml 文件中添加以下权限：

```
<uses-permission android:name="android.permission.RECEIVE_SMS"/><!-- 接收短信权限 -->
<uses-permission android:name="android.permission.SEND_SMS"/><!-- 发送短信权限 -->
```

9.3.4 广播接收者

除了短信到来广播 Intent，Android 还有很多广播 Intent，如：开机启动、电池电量变化、时间已经改变等广播 Intent。

- 接收电池电量变化广播 Intent ，在 AndroidManifest.xml 文件中的<application>节点里订阅此 Intent:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BATTERY_CHANGED"/>
    </intent-filter>
</receiver>
```

- 接收开机启动广播 Intent，在 AndroidManifest.xml 文件中的<application>节点里订阅此 Intent:

```
<receiver android:name=".IncomingSMSReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

并且要进行权限声明：

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

通常一个 BroadcastReceiver 对象的生命周期不超过 5 秒，所以在 BroadcastReceiver 里不能做一些比较耗时的操作，如果需要完成一项比较耗时的工作，可以通过发送 Intent 给 Activity 或 Service，由 Activity 或 Service 来完成。

```
public class IncomingSMSReceiver extends BroadcastReceiver {
    @Override public void onReceive(Context context, Intent intent) {
            //发送 Intent 启动服务，由服务来完成比较耗时的操作
            Intent service = new Intent(context, XxxService.class);
            context.startService(service);
            //发送 Intent 启动 Activity，由 Activity 来完成比较耗时的操作
            Intent newIntent = new Intent(context, XxxActivity.class);
            context.startActivity(newIntent);
    }
}
```

9.3.5 闹钟与提醒服务 Demo

- 研究案例：AlarmDemo
- 研究案例： MultiAlarmReceiver

第十天.Android 网络与通信

10.1 Android 网络通讯介绍

10.1.1 网络通讯技术

- **Java.net**
- **Apache HttpClient**
- **Socket** 技术
- 装载网页
- **WiFi** 技术
- **Bluetooth** 蓝牙

10.2 Java.net

10.2.2 主 Activity

```
public class Activity01 extends Activity{
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main)
        Button button_http = (Button) findViewById(R.id.Button_HTTP);
        /* 监听 button 的事件信息 */
        button_http.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v){
            /* 新建一个 Intent 对象 */
            Intent intent = new Intent();
            /* 指定 intent 要启动的类 */
            intent.setClass(Activity01.this, Activity02.class);
            /* 启动一个新的 Activity */
            startActivity(intent);
            /* 关闭当前的 Activity */
            Activity01.this.finish();
            }
        });
**  Activity02 是直接获取数据的 demo!
Button button_Get = (Button) findViewById(R.id.Button_Get);
        /* 监听 button 的事件信息 */
        button_Get.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v)
            {
                /* 新建一个 Intent 对象 */
                Intent intent = new Intent();
                /* 指定 intent 要启动的类 */
                intent.setClass(Activity01.this, Activity03.class);
                /* 启动一个新的 Activity */
                startActivity(intent);
```

```
                /* 关闭当前的 Activity */
                Activity01.this.finish();
            }
        });
**  Activity03 是以 Get 方式上传参数的 demo!
Button button_Post = (Button) findViewById(R.id.Button_Post);
        /* 监听 button 的事件信息 */
        button_Post.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v)
            {
                /* 新建一个 Intent 对象 */
                Intent intent = new Intent();
                /* 指定 intent 要启动的类 */
                intent.setClass(Activity01.this, Activity04.class);
                /* 启动一个新的 Activity */
                startActivity(intent);
                /* 关闭当前的 Activity */
                Activity01.this.finish();
            }
        });
    }
} //结束 Activity1 类
**  Activity04 是以 Post 方式上传参数的 demo!
```

10.2.3 直接获取数据

```
//http 地址
String httpUrl = "http://serverip:8080/AndroidAppServer/index.jsp";
//获得的数据
String resultData = ""; URL url = null;
//构造一个 URL 对象
url = new URL(httpUrl);
    //使用 HttpURLConnection 打开连接
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
//得到读取的内容(流)
InputStreamReader in = new InputStreamReader(urlConn.getInputStream());
// 为输出创建 BufferedReader
BufferedReader buffer = new BufferedReader(in);
String inputLine = null;
//使用循环来读取获得的数据
while (((inputLine = buffer.readLine()) != null)){
    resultData += inputLine + "\n";
}
**  研究 HttpURLConnectionDemo 工程(Activity02.java)
```

** serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1

10.2.4 以 Get 方式上传参数

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp?par=abcdefg";
    URL url = new URL(httpUrl);
    // 使用 HttpURLConnection 打开连接
    HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
    //得到读取的内容(流)
    InputStreamReader in = new InputStreamReader(urlConn.getInputStream());
    // 为输出创建 BufferedReader
    BufferedReader buffer = new BufferedReader(in);
    String inputLine = null;
    //使用循环来读取获得的数据
    while (((inputLine = buffer.readLine()) != null)){ resultData += inputLine + "\n"; }
    //关闭 InputStreamReader
    in.close();
    //关闭 http 连接
    urlConn.disconnect();
** 研究 HttpURLConnectionDemo 工程(Activity03.java)
**  serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1
```

10.2.5 以 Post 方式上传参数

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp";
URL url = new URL(httpUrl);
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
//因为这个是 post 请求,设立需要设置为 true
urlConn.setDoOutput(true); urlConn.setDoInput(true);
urlConn.setRequestMethod("POST");          // 设置以 POST 方式
urlConn.setUseCaches(false);               // Post 请求不能使用缓存
urlConn.setInstanceFollowRedirects(true);
urlConn.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
urlConn.connect();
DataOutputStream out = new DataOutputStream(urlConn.getOutputStream());
String content = "par=" + URLEncoder.encode("ABCDEFG", "gb2312"); //要上传的参数
out.writeBytes(content);                   //将要上传的内容写入流中
out.flush();          out.close();
//获取数据
```

```
BufferedReader reader = new BufferedReader(new
InputStreamReader(urlConn.getInputStream()));
String inputLine = null;
while (((inputLine = reader.readLine()) != null)){resultData += inputLine + "\n";}
    reader.close(); urlConn.disconnect();
**  研究 HttpURLConnectionDemo 工程(Activity03.java)
**   serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1

main.xml

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="通过下面的按钮进行不同方式的连接"/>
    <Button
        android:id="@+id/Button_HTTP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="直接获取数据"/>
    <Button
        android:id="@+id/Button_Get"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="以 GET 方式传递数据"/>
    <Button
        android:id="@+id/Button_Post"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="以 POST 方式传递数据"/>
</LinearLayout>


AndroidManifest.xml

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.lxt008"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
```

```
        <activity android:name=".Activity01"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Activity02"></activity>
        <activity android:name="Activity03"></activity>
        <activity android:name="Activity04"></activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

10.3 Apache HttpClient

10.3.1 使用 HttpClient:主 Activity

```
Button button_Get = (Button) findViewById(R.id.Button_Get);
button_Get.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v){
        Intent intent = new Intent(); /* 新建一个Intent对象 */
        /* 指定intent要启动的类 */
        intent.setClass(Activity01.this, Activity02.class);
        startActivity(intent); /* 启动一个新的Activity */
        Activity01.this.finish(); /* 关闭当前的Activity */ }
        });

Button button_Post = (Button) findViewById(R.id.Button_Post);
button_Post.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v){
        Intent intent = new Intent();
        /* 指定intent要启动的类 */
        intent.setClass(Activity01.this, Activity03.class);
        startActivity(intent);
        Activity01.this.finish();}
        });
    }

 ** 研究ApacheHttpClientDemo工程(Activity01.java)
 ** serverip要换成真实IP,不能用localhost或127.0.0.1
Button button_Get = (Button) findViewById(R.id.Button_Get);
button_Get.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v){
        Intent intent = new Intent(); /* 新建一个 Intent 对象 */
```

```
        /* 指定 intent 要启动的类 */
        intent.setClass(Activity01.this, Activity02.class);
        startActivity(intent); /* 启动一个新的 Activity */
    Activity01.this.finish(); /* 关闭当前的 Activity */ }
        });

Button button_Post = (Button) findViewById(R.id.Button_Post);
button_Post.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v){
        Intent intent = new Intent();
        /* 指定 intent 要启动的类 */
        intent.setClass(Activity01.this, Activity03.class);
        startActivity(intent);                    Activity01.this.finish();}
        });
    }
**   研究 ApacheHttpClientDemo 工程(Activity01.java)
**   serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1
```

10.3.2 HttpClient:HttpGet

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp?par=HttpClient_android_Get";
//HttpGet连接对象
HttpGet httpRequest = new HttpGet(httpUrl);
//取得HttpClient对象
HttpClient httpclient = new DefaultHttpClient();
//请求HttpClient，取得HttpResponse
HttpResponse httpResponse = httpclient.execute(httpRequest);
//请求成功
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
    //取得返回的字符串
    String strResult = EntityUtils.toString(httpResponse.getEntity());
    mTextView.setText(strResult);
}
else{
    mTextView.setText("请求错误!");
}
** 研究ApacheHttpClientDemo工程(Activity02.java)
** serverip要换成真实IP,不能用localhost或127.0.0.1
```

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp?par=HttpClient_android_Get";
//HttpGet 连接对象
HttpGet httpRequest = new HttpGet(httpUrl);
//取得 HttpClient 对象
HttpClient httpclient = new DefaultHttpClient();
```

```
//请求 HttpClient，取得 HttpResponse
HttpResponse httpResponse = httpclient.execute(httpRequest);
//请求成功
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
    //取得返回的字符串
    String strResult = EntityUtils.toString(httpResponse.getEntity());
    mTextView.setText(strResult);
}
else{
    mTextView.setText("请求错误!");
}
**   研究 ApacheHttpClientDemo 工程(Activity02.java)
**   serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1
```

10.3.3 HttpClient:HttpPost

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp";
//HttpPost连接对象
HttpPost httpRequest = new HttpPost(httpUrl);
//使用NameValuePair来保存要传递的Post参数
List<NameValuePair> params = new ArrayList<NameValuePair>();
//添加要传递的参数
params.add(new BasicNameValuePair("par", "HttpClient_android_Post"));
//设置字符集
HttpEntity httpentity = new UrlEncodedFormEntity(params, "gb2312");
//请求httpRequest
httpRequest.setEntity(httpentity);
//取得默认的HttpClient
HttpClient httpclient = new DefaultHttpClient();
//取得HttpResponse
HttpResponse httpResponse = httpclient.execute(httpRequest);
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
    String strResult = EntityUtils.toString(httpResponse.getEntity());}

** 研究ApacheHttpClientDemo工程(Activity03.java)
** serverip要换成真实IP,不能用localhost或127.0.0.1
```

```
String httpUrl = "http://192.168.1.110:8080/httpget.jsp";
//HttpPost 连接对象
HttpPost httpRequest = new HttpPost(httpUrl);
//使用 NameValuePair 来保存要传递的 Post 参数
List<NameValuePair> params = new ArrayList<NameValuePair>();
//添加要传递的参数
params.add(new BasicNameValuePair("par", "HttpClient_android_Post"));
//设置字符集
HttpEntity httpentity = new UrlEncodedFormEntity(params, "gb2312");
//请求 httpRequest
```

```
httpRequest.setEntity(httpentity);
//取得默认的 HttpClient
HttpClient httpclient = new DefaultHttpClient();
//取得 HttpResponse
HttpResponse httpResponse = httpclient.execute(httpRequest);
if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
    String strResult = EntityUtils.toString(httpResponse.getEntity());}
**   研究 ApacheHttpClientDemo 工程(Activity03.java)
**   serverip 要换成真实 IP,不能用 localhost 或 127.0.0.1
```

10.4 装载并显示 Web 网页

10.4.1 用线程刷新网页显示

```
public void onCreate(Bundle savedInstanceState){ new Thread(mRunnable).start(); //开启线程 }
    //刷新网页显示
    private void refresh(){
            URL url = new URL("http://192.168.1.110:8080/date.jsp");
            HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
            InputStreamReader in = new InputStreamReader(urlConn.getInputStream());
            BufferedReader buffer = new BufferedReader(in);
            String inputLine = null;
            // 使用循环来读取获得的数据 // 关闭InputStreamReader // 设置显示取得的内容 }
}
private Runnable mRunnable = new Runnable() {
    public void run() {
       while (true) {      Thread.sleep(5 * 1000);
         //发送消息
         mHandler.sendMessage(mHandler.obtainMessage()); };
     Handler mHandler = new Handler() {
      public void handleMessage(Message msg) {
       super.handleMessage(msg);
       refresh();
      } };
** 研究HttpURLConnectionRefresh工程
```

```
public void onCreate(Bundle savedInstanceState){ new Thread(mRunnable).start(); //开启线
程 }
    //刷新网页显示
    private void refresh(){
        URL url = new URL("http://192.168.1.110:8080/date.jsp");
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        InputStreamReader in = new InputStreamReader(urlConn.getInputStream());
        BufferedReader buffer = new BufferedReader(in);
        String inputLine = null;
        // 使用循环来读取获得的数据 // 关闭 InputStreamReader // 设置显示取得的内
容 }
}
```

```
private Runnable mRunnable = new Runnable()   {
      public void run()   {
            while (true) {              Thread.sleep(5 * 1000);
                //发送消息
                mHandler.sendMessage(mHandler.obtainMessage());   };
        Handler mHandler = new Handler() {
          public void handleMessage(Message msg)   {
            super.handleMessage(msg);
            refresh();
          } };
**     研究 HttpURLConnectionRefresh 工程
```

10.4.2 装载网页并显示

```
WebView webView = (WebView) findViewById(R.id.webview);
String html = "<html><body>Hello lxt008!!!</body></html>";

//装载页面,你可以换成URL地址。
webView.loadDataWithBaseURL("Demo", html, "text/html", "utf-8", null);
//设置支持JS
webView.getSettings().setJavaScriptEnabled(true);
//设置浏览器客户端
webView.setWebChromeClient(new WebChromeClient());


<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical" android:layout_width="fill_parent"
      android:layout_height="fill_parent">
      <WebView android:id="@+id/webview" android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
</LinearLayout>
```

```
WebView webView = (WebView) findViewById(R.id.webview);
String html = "<html><body>Hello lxt008!!!</body></html>";
//装载页面,你可以换成 URL 地址。
webView.loadDataWithBaseURL("Demo", html, "text/html", "utf-8", null);
//设置支持 JS
webView.getSettings().setJavaScriptEnabled(true);
//设置浏览器客户端
  webView.setWebChromeClient(new WebChromeClient());
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical" android:layout_width="fill_parent"
      android:layout_height="fill_parent">
      <WebView android:id="@+id/webview" android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent" />
</LinearLayout>
```

10.5 Socket 编程复习

- 以前课程中学过 **Socket** 编程。
- 研究 **SocketDemo** 以复习巩固 **Socket** 在 **Android** 中的使用。

第十一天.Android 图形技术

11.1 Paint 类与 Canvas 类

11.1.1 绘图 Paint 类

```
Paint mPaint = new Paint();
/* 设置 Paint 为无锯齿 */
mPaint.setAntiAlias(true);
/* 设置 Paint 的颜色 */
mPaint.setColor(Color.RED);
mPaint.setColor(Color.BLUE);
/* 同样是设置颜色 */
mPaint.setColor(Color.rgb(255, 0, 0));
/* 提取颜色 */
Color.red(0xcccccc);
Color.green(0xcccccc);
/* 设置 paint 的颜色和 Alpha 值(a,r,g,b) */
mPaint.setARGB(255, 255, 0, 0);
/* 设置 paint 的 Alpha 值 */
mPaint.setAlpha(220);


/* 设置字体的尺寸 */
mPaint.setTextSize(14);
// 设置 paint 的风格为“空心”.
// 当然也可以设置为“实心”(Paint.Style.FILL)
mPaint.setStyle(Paint.Style.STROKE);
```

```
// 设置“空心”的外框的宽度。
mPaint.setStrokeWidth(5);
/* 得到 Paint 的一些属性 */
Log.i(TAG, "paint 的颜色：" + mPaint.getColor());
Log.i(TAG, "paint 的 Alpha：" + mPaint.getAlpha());
Log.i(TAG, "paint 的外框的宽度：" + mPaint.getStrokeWidth());
Log.i(TAG, "paint 的字体尺寸：" + mPaint.getTextSize());
/* 绘制一个矩形 */
canvas.drawRect((320 - 80) / 2, 20, (320 - 80) / 2 + 80, 20 + 40, mPaint);
/* 设置风格为实心 */
mPaint.setStyle(Paint.Style.FILL);
mPaint.setColor(Color.GREEN);
/* 绘制绿色实心矩形 */
canvas.drawRect(0, 20, 40, 20 + 40, mPaint);
```

11.1.2 在线程中更新界面

```
public void run()  {
        while (!Thread.currentThread().isInterrupted())
        {
            try{
                Thread.sleep(100);
            }
            catch (InterruptedException e){
                Thread.currentThread().interrupt();
            }
            // 使用 postInvalidate 可以直接在线程中更新界面
            postInvalidate(); //会调用 onDraw(Canvas canvas)方法
        }
    }
** 研究案例 PaintDemo
```

11.1.3 Canvas 画布类

```
public void onDraw(Canvas canvas){
        super.onDraw(canvas);
        /* 设置画布的颜色 */
        canvas.drawColor(Color.BLACK);
        /* 设置取消锯齿效果 */
        mPaint.setAntiAlias(true);
```

```
        /* 设置裁剪区域 */
        canvas.clipRect(10, 10, 280, 260);
        /* 线锁定画布 */
        canvas.save();
        /* 旋转画布 */
        canvas.rotate(45.0f);
        /* 设置颜色及绘制矩形 */
        mPaint.setColor(Color.RED);
        canvas.drawRect(new Rect(15,15,140,70), mPaint);
        /* 解除画布的锁定 */
        canvas.restore();
        /* 设置颜色及绘制另一个矩形 */
        mPaint.setColor(Color.GREEN);
        canvas.drawRect(new Rect(150,75,260,120), mPaint);
    }
```

** 研究案例 **PaintDemo2**

11.2 SurfaceView 类

11.2.1 SurfaceView 类

- 执行效率高
- 可以直接访问画布 **Canvas**
- 开发手机游戏经常用

11.2.2 SurfaceView 使用要点

1.public class GameSurfaceView extends SurfaceView implements SurfaceHolder.Callback,Runnable

2.定义 **SurfaceHolder** 对象：

```
SurfaceHolder mSurfaceHolder= this.getHolder();
```

3.添加回调：

```
mSurfaceHolder.addCallback(this);
this.setFocusable(true);
```

4.在回调方法 **surfaceCreated()**中开启绘图线程。

5.在绘图线程中绘制图形。

11.2.3 SurfaceView 回调方法

```
// 在 surface 的大小发生改变时激发
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }
    // 在 surface 创建时激发
    public void surfaceCreated(SurfaceHolder holder){
        //开启绘图线程
        new Thread(this).start();
    }
    // 在 surface 销毁时激发
    public void surfaceDestroyed(SurfaceHolder holder){
        // 停止循环
        mbLoop = false;
    }
```

11.2.3 绘图线程

```
// 绘图循环
    public void run(){
        while (mbLoop){
            try{
                Thread.sleep(200);
            }
            catch (Exception e){

            }
            synchronized( mSurfaceHolder ){
                draw();
            }

        }
    }
```

11.2.4 绘图方法

```
public void draw() {
    //锁定画布，得到 canvas
    Canvas canvas= mSurfaceHolder.lockCanvas();
    if (mSurfaceHolder==null || canvas == null ){
```

```
            return;
        }
        // 绘图
        Paint mPaint = new Paint();
        mPaint.setColor(Color.BLACK);
        //绘制矩形--清屏作用
        canvas.drawRect(0, 0, 320, 480, mPaint);
        mPaint.setColor(Color.BLUE);
        // 绘制矩形
        canvas.drawCircle((320 - 25) / 2, y, 50, mPaint);
        // 绘制后解锁，绘制后必须解锁才能显示
        mSurfaceHolder.unlockCanvasAndPost(canvas);
    }
```

** 研究完整案例 **GameSurfaceViewDemo**

11.3 绘制几何形状

11.3.1 绘制几何形状

```
/* 定义矩形对象 */
    Rect rect1 = new Rect();
    /* 设置矩形大小 */
    rect1.left = 5; rect1.top = 5; rect1.bottom = 25; rect1.right = 45;
    mPaint.setColor(Color.BLUE);
    /* 绘制矩形 */
    canvas.drawRect(rect1, mPaint);
    /* 绘制矩形 */
    canvas.drawRect(50, 5, 90, 25, mPaint);

    /* 绘制圆形(圆心 x,圆心 y,半径 r,p) */
    canvas.drawCircle(40, 70, 30, mPaint);

    /* 定义椭圆对象 */
    RectF rectf1 = new RectF();
    /* 设置椭圆大小 */
    rectf1.left = 80; rectf1.top = 30; rectf1.right = 120; rectf1.bottom = 70;
    /* 绘制椭圆 */
    canvas.drawOval(rectf1, mPaint);

/* 绘制多边形 */
    Path path1 = new Path();
    /*设置多边形的点*/
```

```
        path1.moveTo(150+5, 80-50);
        path1.lineTo(150+45, 80-50);
        path1.lineTo(150+30, 120-50);
        path1.lineTo(150+20, 120-50);
        /* 使这些点构成封闭的多边形 */
        path1.close();

        mPaint.setColor(Color.GRAY);
        /* 绘制这个多边形 */
        canvas.drawPath(path1, mPaint);
        mPaint.setColor(Color.RED);
        mPaint.setStrokeWidth(3);
        /* 绘制直线 */
        canvas.drawLine(5, 110, 315, 110, mPaint);
```

** 研究完整案例 **PaintDemo3**

11.3.2 ShapeDrawable 绘制几何图形

```
/* 实例化 ShapeDrawable 对象并说明是绘制一个矩形 */
            ShapeDrawable    mShapeDrawable = new ShapeDrawable(new RectShape());

            //得到画笔 paint 对象并设置其颜色
            mShapeDrawable.getPaint().setColor(Color.RED);
            Rect bounds = new Rect(5, 250, 55, 280);
            /* 设置图像显示的区域 */
            mShapeDrawable.setBounds(bounds);
            mShapeDrawable.draw(canvas); /* 绘制图像 */

            /* 实例化 ShapeDrawable 对象并说明是绘制一个椭圆 */
            mShapeDrawable = new ShapeDrawable(new OvalShape());
            //得到画笔 paint 对象并设置其颜色
            mShapeDrawable.getPaint().setColor(Color.GREEN);
            /* 设置图像显示的区域 */
            mShapeDrawable.setBounds(70, 250, 150, 280);
            /* 绘制图像 */
            mShapeDrawable.draw(canvas);


Path path1 = new Path();
        /*设置多边形的点*/
        path1.moveTo(150+5, 80+80-50);
        path1.lineTo(150+45, 80+80-50);
        path1.lineTo(150+30, 80+120-50);
```

```
path1.lineTo(150+20, 80+120-50);
/* 使这些点构成封闭的多边形 */
path1.close();
//PathShape 后面两个参数分别是宽度和高度
mShapeDrawable = new ShapeDrawable(new PathShape(path1,150,150));
//得到画笔 paint 对象并设置其颜色
mShapeDrawable.getPaint().setColor(Color.BLUE);
/* 设置图像显示的区域 */
mShapeDrawable.setBounds(100, 170, 200, 280);
/* 绘制图像 */
mShapeDrawable.draw(canvas);
```

** 研究案例 PaintDemo3 中 GameView2.java

11.4 图形绘制与旋转缩放

11.4.1 绘制图像 1

```
/* 从资源文件中装载图片 */
//getResources()->得到 Resources
//getDrawable()->得到资源中的 Drawable 对象，参数为资源索引 ID
//getBitmap()->得到 Bitmap
        Bitmap mBitQQ = ((BitmapDrawable) getResources().getDrawable(R.drawable.qq)).getBitmap();

/* 清屏效果 */
canvas.drawColor(Color.GRAY);
/* 在屏幕(0,0)处绘制图片 mBitQQ */
GameView.drawImage(canvas, mBitQQ, 0, 0);
```

** drawImage()参考后面 PPT

11.4.2 绘制图像 2

```
* @param    x 屏幕上的 x 坐标
 * @param    y 屏幕上的 y 坐标
 * @param    w 要绘制的图片的宽度
 * @param    h 要绘制的图片的高度
 * @param    bx 图片上的 x 坐标
 * @param    by 图片上的 y 坐标
```

```
    public static void drawImage(Canvas canvas, Bitmap blt, int x, int y, int w, int h, int bx,
int by)   {
        Rect src = new Rect();// 图片
        Rect dst = new Rect();// 屏幕
        src.left = bx; src.top = by; src.right = bx + w;
        src.bottom = by + h;
        dst.left = x; dst.top = y; dst.right = x + w;
        dst.bottom = y + h;
        canvas.drawBitmap(blt, src, dst, null);
        src = null; dst = null;
    }
```

11.4.3 绘制图像 3

```
    /**
     * 绘制一个 Bitmap
     * @param canvas      画布
     * @param bitmap      图片
     * @param x       屏幕上的 x 坐标
     * @param y       屏幕上的 y 坐标
     */
    public static void drawImage(Canvas canvas, Bitmap bitmap, int x, int y)
    {
        /* 绘制图像 */
        canvas.drawBitmap(bitmap, x, y, null);
    }
** 还需要使用线程更新界面
** 研究案例 PaintDemo5
```

11.4.5 图像旋转

```
public void onDraw(Canvas canvas){
        super.onDraw(canvas);
        Matrix mMatrix = new Matrix();
        /* 重置 mMatrix */
        mMatrix.reset();
        /* 设置旋转 */
        mMatrix.setRotate(角度);

        /* 按 mMatrix 得旋转构建新的 Bitmap */
        Bitmap mBitQQ2 = Bitmap.createBitmap(mBitQQ, 0, 0, BitQQwidth,BitQQheight,
mMatrix, true);
```

```
        /* 绘制旋转之后的图片 */
        GameView.drawImage(canvas, mBitQQ2, (320-BitQQwidth)/2, 10);
        mBitQQ2 = null;
    }
```

** 研究 **MatrixDemo**

11.4.6 图像缩放

```
public void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        /* 重置 mMatrix */
        mMatrix.reset();
        /* 设置缩放 */
        mMatrix.postScale(Scale,Scale);

        /* 按 mMatrix 得旋转构建新的 Bitmap */
        Bitmap mBitQQ2 = Bitmap.createBitmap(mBitQQ, 0, 0, BitQQwidth,BitQQheight,
mMatrix, true);
        /* 绘制旋转之后的图片 */
        GameView.drawImage(canvas, mBitQQ2, (320-BitQQwidth)/2, 10);

        mBitQQ2 = null;
    }
```

** 研究 **MatrixScaleDemo**

11.5 用 Shader 类进行渲染

- BitmapShader： Bitmap 渲染
- LinearGradient： 线性渐变渲染
- ComposeShader： 混合渲染
- RadialGradient： 环形渐变渲染
- SweepGradient： 梯度渲染

** 研究案例 **ShaderDemo**

第十二天.Android 动画技术

12.1 Tween 动画

12.1.1 动画实现

- Tween 动画
  - 对场景中的对象不断进行图像变换，如平移、缩放、旋转。
  - Frame 帧动画
  - 顺序播放事先做好的图像，如电影。
  - GIF 动画

12.1.2 代码实现 Tween 动画 1

```
/* 装载资源 */
Bitmap mBitQQ mBitQQ = ((BitmapDrawable) getResources().getDrawable(R.drawable.qq)).getBitmap();
/* 绘制图片 */
canvas.drawBitmap(mBitQQ, 0, 0, null);

/* 创建 Alpha 动画 */
private Animation mAnimationAlpha = new AlphaAnimation(0.1f, 1.0f);
/* 设置动画的时间 */
mAnimationAlpha.setDuration(3000);
/* 开始播放动画 */
this.startAnimation(mAnimationAlpha);

/* 创建 Scale 动画 */
private Animation mAnimationScale =new ScaleAnimation(0.0f, 1.0f, 0.0f, 1.0f,Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
/* 设置动画的时间 */
mAnimationScale.setDuration(500);
/* 开始播放动画 */
this.startAnimation(mAnimationScale);
```

12.1.3 代码实现 Tween 动画 2

```
/* 创建 Translate 动画 */
private Animation mAnimationTranslate = new TranslateAnimation(10, 100,10, 100);
/* 设置动画的时间 */
```

```
mAnimationTranslate.setDuration(1000);
/* 开始播放动画 */
this.startAnimation(mAnimationTranslate);

/* 创建 Rotate 动画 */
private Animation mAnimationRotate=new RotateAnimation(0.0f,
+360.0f,Animation.RELATIVE_TO_SELF,0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
/* 设置动画的时间 */
mAnimationRotate.setDuration(1000);
/* 开始播放动画 */
this.startAnimation(mAnimationRotate);
```

12.2.4 代码实现 Tween 动画:main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"    >
<TextView
    android:layout_width="fill_parent"    android:layout_height="wrap_content"
    android:text="@string/hello"    />
    <Button
    android:id="@+id/AlphaAnimation"    android:layout_width="fill_parent"
    android:layout_height="wrap_content"    android:text="Alpha 动画"/>
    <Button
    android:id="@+id/ScaleAnimation"    android:layout_width="fill_parent"
    android:layout_height="wrap_content"    android:text="Scale 动画"/>
    <Button
    android:id="@+id/TranslateAnimation"    android:layout_width="fill_parent"
    android:layout_height="wrap_content"    android:text="Translate 动画"/>
    <Button
    android:id="@+id/RotateAnimation"    android:layout_width="fill_parent"
    android:layout_height="wrap_content"    android:text="Rotate 动画"/>
</LinearLayout>
```

12.2.5 XML 布局实现 Tween 动画

```
/* 装载动画布局 */
mAnimationAlpha = AnimationUtils.loadAnimation(mContext,R.anim.alpha_animation);
/* 开始播放动画 */
this.startAnimation(mAnimationAlpha);
/* 装载动画布局 */
mAnimationScale = AnimationUtils.loadAnimation(mContext,R.anim.scale_animation);
```

```
this.startAnimation(mAnimationScale);
/* 装载动画布局 */
mAnimationTranslate =
AnimationUtils.loadAnimation(mContext,R.anim.translate_animation);
this.startAnimation(mAnimationTranslate);
/* 装载动画布局 */
mAnimationRotate = AnimationUtils.loadAnimation(mContext,R.anim.rotate_animation);
this.startAnimation(mAnimationRotate);
```

R.anim.alpha_animation

```
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<alpha
    android:fromAlpha="0.1"
    android:toAlpha="1.0"
    android:duration="2000"
/>
</set>
```

R.anim.scale_animation

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
            android:interpolator="@android:anim/accelerate_decelerate_interpolator"

            android:fromXScale="0.0"
            android:toXScale="1.0"

            android:fromYScale="0.0"
            android:toYScale="1.0"

            android:pivotX="50%"
            android:pivotY="50%"

            android:fillAfter="false"
            android:duration="500" />
</set>
```

R.anim.translate_animation

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
<translate
```

```
        android:fromXDelta="10"
        android:toXDelta="100"
        android:fromYDelta="10"
        android:toYDelta="100"
        android:duration="1000"
/>
</set>


R.anim.rotate_animation

<set xmlns:android="http://schemas.android.com/apk/res/android">
<rotate
android:interpolator="@android:anim/accelerate_decelerate_interpolator"

            android:fromDegrees="0"
            android:toDegrees="+360"

            android:pivotX="50%"
            android:pivotY="50%"

            android:duration="1000" />
</set>


** 案例 AnimationDemo2
```

12.2 Frame 帧动画

12.2.1 代码实现 Frame 动画

```
/* 实例化 AnimationDrawable 对象 */
      private AnimationDrawable frameAnimation = new AnimationDrawable();
      /* 装载资源 */
      //这里用一个循环了装载所有名字类似的资源，如“a1.....15.png”的图片
      for (int i = 1; i <= 15; i++){
            int id = getResources().getIdentifier("a" + i, "drawable",
                  mContext.getPackageName());
             Drawable mBitAnimation = getResources().getDrawable(id);
            /* 为动画添加一帧 */
            //参数 mBitAnimation 是该帧的图片
```

```
            //参数 500 是该帧显示的时间,按毫秒计算
            frameAnimation.addFrame(mBitAnimation, 500);
        }
        /*  设置播放模式是否循环 false 表示循环而 true 表示不循环  */
        frameAnimation.setOneShot( false );
        /*  设置本类将要显示这个动画  */
        this.setBackgroundDrawable(frameAnimation);
        /*  开始播放动画  */
        frameAnimation.start();
```

** 案例 AnimationDrawableDemo

12.2.2 XML 实现 Frame 动画

```
/*  定义 AnimationDrawable 动画对象  */
private AnimationDrawable frameAnimation = null;
/*  定义一个 ImageView 用来显示动画  */
ImageView img = new ImageView(mContext);
/*  装载动画布局文件  */
img.setBackgroundResource(R.anim.frameanimation);
/*  构建动画  */
private AnimationDrawable frameAnimation = (AnimationDrawable) img.getBackground();
/*  设置是否循环  */
frameAnimation.setOneShot( false );
/*  设置该类显示的动画  */
this.setBackgroundDrawable(frameAnimation);
/*  开始播放动画  */
frameAnimation.start();
```

frameanimation.xml

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
        android:oneshot="false">
        <item android:drawable="@drawable/a1" android:duration="500" />
        <item android:drawable="@drawable/a2" android:duration="500" />
        <item android:drawable="@drawable/a3" android:duration="500" />
        <item android:drawable="@drawable/a4" android:duration="500" />
        <item android:drawable="@drawable/a5" android:duration="500" />
        <item android:drawable="@drawable/a6" android:duration="500" />
        <item android:drawable="@drawable/a7" android:duration="500" />
```

```
    <item android:drawable="@drawable/a8" android:duration="500" />
    <item android:drawable="@drawable/a9" android:duration="500" />
    <item android:drawable="@drawable/a10" android:duration="500" />
    <item android:drawable="@drawable/a11" android:duration="500" />
    <item android:drawable="@drawable/a12" android:duration="500" />
    <item android:drawable="@drawable/a13" android:duration="500" />
    <item android:drawable="@drawable/a14" android:duration="500" />
    <item android:drawable="@drawable/a15" android:duration="500" />
</animation-list>
```

** 案例 AnimationDrawableDemo2

12.3 GIF 动画

- 简单介绍案例 GifAnimationDemo

12.4 全屏与横屏技术

```
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);

    /* 设置为无标题栏 */
    requestWindowFeature(Window.FEATURE_NO_TITLE);

    /* 设置为全屏模式 */
        getWindow().setFlags(
          WindowManager.LayoutParams.FLAG_FULLSCREEN,
    WindowManager.LayoutParams.FLAG_FULLSCREEN );

    /* 设置为横屏 */
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
      setContentView(R.layout.main);
    }
```

** 全屏技术在拍照、录制视频、游戏中很常用

12.5 获取屏幕属性

```
public void onCreate(Bundle savedInstanceState){
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 定义 DisplayMetrics 对象 */
        DisplayMetrics dm = new DisplayMetrics();
        /* 取得窗口属性 */
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        /* 窗口的宽度 */
        int screenWidth = dm.widthPixels;
        /* 窗口的高度 */
        int screenHeight = dm.heightPixels;
        mTextView = (TextView) findViewById(R.id.TextView01);
        mTextView.setText("屏幕宽度:" + screenWidth + "\n 屏幕高度:" + screenHeight);
    }
```

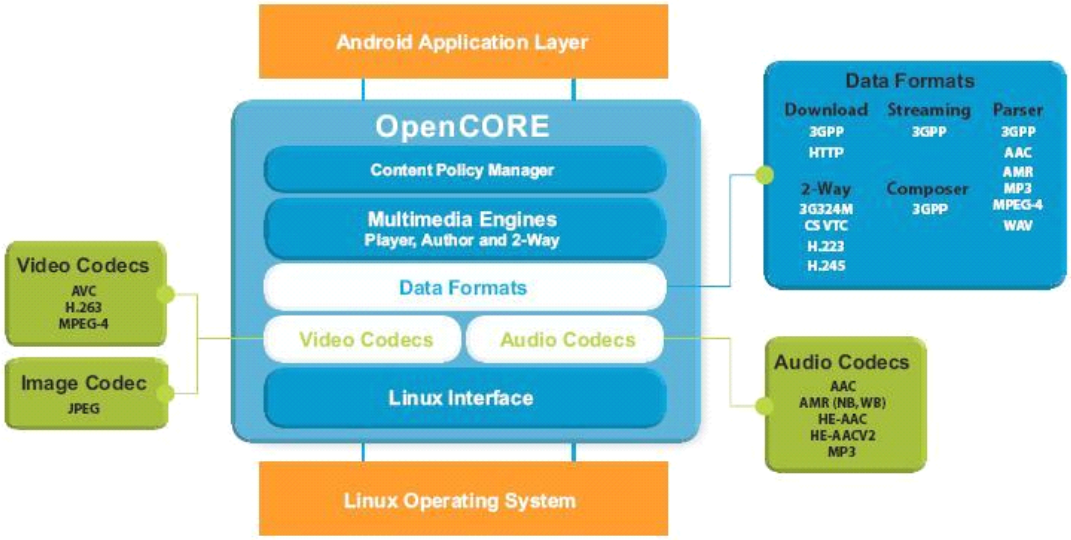
第十三天.Android 多媒体开发

13.1 播放音乐

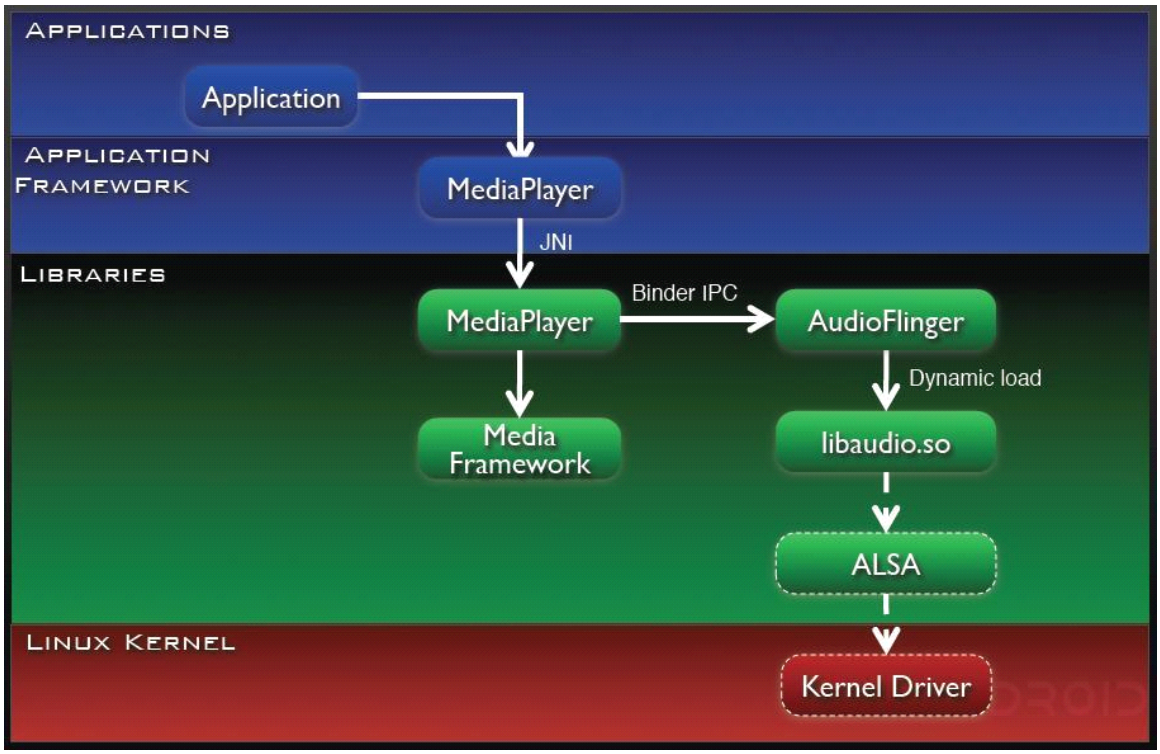
13.1.1 多媒体架构

- 基于第三方 **Packet Video** 公司的 **Open Core platform** 实现
- 支持几乎所有通用的音频、视频、静态图像格式,包括：**MPEG4**、**H.264**、**MP3**、**AAC**、**AMR**、**JPG**、**PNG**、**GIF** 等。
- 通过 **Open Core** 程序员可以方便快速的开发出想要的多媒体应用程序，例如：音视频的采集，回放，视频会议，实时的流媒体播放等应用。
- **Android** 提供了 **MediaPlayer** 和 **MediaRecorder** 等上层的 **Media API** 给开发人员使用。
- **Open Core** 的代码在 **Android** 代码的 **External/Opencore** 目录中。

13.1.2 Open Core 框架



13.1.3 调用层次关系



13.1.4 音乐播放

MediaPlayer mediaPlayer = new MediaPlayer();

```
if (mediaPlayer.isPlaying()) {
    mediaPlayer.reset();      //重置为初始状态
}
mediaPlayer.setDataSource("/sdcard/good.mp3");
mediaPlayer.prepare();//缓冲
mediaPlayer.start();          //开始或恢复播放
mediaPlayer.pause();          //暂停播放
mediaPlayer.start();          //恢复播放
mediaPlayer.stop();           //停止播放
mediaPlayer.release();  //释放资源
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            public void onCompletion(MediaPlayer arg0) { //播放完毕调用
            mediaPlayer.release();
            } });
mediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener() { //  错误处理事件
             public boolean onError(MediaPlayer player, int arg1, int arg2) {
      mediaPlayer.release();
      return false;
            } });
```

13.2 播放视频

13.2.1 播放视频

```
/*  创建 VideoView 对象  */
final VideoView videoView = (VideoView) findViewById(R.id.VideoView01);
/*设置视频路径*/
videoView.setVideoPath("/sdcard/test.mp4");
/*  设置模式-播放进度条  */
videoView.setMediaController(new MediaController(Activity01.this));
videoView.requestFocus();
/*  开始播放  */
videoView.start();
/*  暂停  */
videoView.pause();
```

13.2.2 音乐/视频播放案例

- 音乐播放案例

- **MediaPlayerDemo**
- 视频播放案例
- **VideoViewDemo**

13.3 录制音频

13.3.1 实现录音功能

第一步：在功能清单文件 **AndroidManifest.xml** 中添加录音与写 **sdcard** 权限：

```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

第二步：编写音频刻录代码：

```
MediaRecorder recorder = new MediaRecorder();
 recorder.setAudioSource(MediaRecorder.AudioSource.MIC);//从麦克风采集声音
 recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);//内容输出格式
 recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);//音频编码方式
 recorder.setOutputFile("/sdcard/lxt008.amr");
 recorder.prepare();          //预期准备
 recorder.start();            //开始刻录
 ...
 recorder.stop();        //停止刻录
 recorder.reset();            //重设
 recorder.release();          //刻录完成一定要释放资源
```

13.3.2 音视频采集

第一步：在功能清单文件 **AndroidManifest.xml** 中添加照相机权限：

```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CAMERA"/>
```

第二步：编写拍照代码：

```
WindowManager wm = (WindowManager) getSystemService(Context.WINDOW_SERVICE);//获取窗口服务
Display display = wm.getDefaultDisplay();//获取屏幕信息
recorder = new MediaRecorder();
recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA); //从照相机采集视频
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
recorder.setVideoSize(display.getWidth(), display.getHeight()); //大小为屏幕的宽和高
recorder.setVideoFrameRate(3); //每秒 3 帧
recorder.setVideoEncoder(MediaRecorder.VideoEncoder.H263); //设置视频编码方式
```

```
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
recorder.setOutputFile("/sdcard/lxt008.3gp");
 recorder.prepare();//预期准备
 recorder.start();     //开始刻录
 ...
 recorder.stop();//停止刻录
 recorder.reset();     //重设
 recorder.release(); //刻录完成一定要释放资源
```

13.4 拍摄照片

13.4.1 录音/拍照案例

13.5 铃声设置

13.5.1 铃声设置：设置各种铃声

```
//打开系统铃声设置
Intent intent = new Intent(RingtoneManager.ACTION_RINGTONE_PICKER);
//设置铃声类型和 title
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE,
RingtoneManager.TYPE_RINGTONE);
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, “设置来电铃声");
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE,
RingtoneManager.TYPE_ALARM);
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "设置闹铃铃声");
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE,
RingtoneManager.TYPE_NOTIFICATION);
intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "设置通知铃声");
//当设置完成之后返回到当前的 Activity
startActivityForResult(intent, ButtonAlarm);
```

13.5.2 铃声设置：回调函数

```
/*  当设置铃声之后的回调函数  */
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
//得到我们选择的铃声
Uri pickedUri = data.getParcelableExtra(RingtoneManager.EXTRA_RINGTONE_PICKED_URI);
//将我们选择的铃声设置成为默认
RingtoneManager.setActualDefaultRingtoneUri(Activity01.this,
RingtoneManager.TYPE_RINGTONE, pickedUri);
Uri pickedUri = RingtoneManager.setActualDefaultRingtoneUri(Activity01.this,
RingtoneManager.TYPE_ALARM, pickedUri);
RingtoneManager.setActualDefaultRingtoneUri(Activity01.this,
RingtoneManager.TYPE_NOTIFICATION, pickedUri);
}
```

13.5.3 铃声设置：main.xml

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"  android:layout_height="fill_parent"  >
  <TextView
  android:id="@+id/TextView01"
  android:layout_width="fill_parent"  android:layout_height="wrap_content"
  android:text="选择下面按钮来设置铃声"  />
  <Button
  android:id="@+id/ButtonRingtone"
  android:layout_width="fill_parent"  android:layout_height="wrap_content"
  android:text="设置来电铃声"  />
  <Button
  android:id="@+id/ButtonAlarm"
  android:layout_width="fill_parent"  android:layout_height="wrap_content"
  android:text="设置闹钟铃声"  />
  <Button
  android:id="@+id/ButtonNotification"
  android:layout_width="fill_parent"  android:layout_height="wrap_content"
  android:text="设置通知铃声"  />
</LinearLayout>
```

13.5.4 铃声设置：AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.yarin.android.Examples_07_08"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Activity01"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

** 研究案例 RingtoneManagerDemo

14. Android 项目案例: mp3 播放器

14.1 需求列表

- 1.创建 Web 应用,管理 mp3 文件。
- 2.编写 XML 文件,其中包含 mp3 文件名。
- 3.Android 程序需要通过网络下载 XML 格式列表文件。
- 4.解析 XML 文件,在手机中显示资源列表。
- 5.点击 mp3 名字需要下载相应 mp3 文件。
- 6.需要支持多线程下载。
- 7.需要显示软件下载的进度。
- 8.能够本地与远程播放 mp3 音乐文件。
- 9.在后台服务中播放 mp3 文件。
- 10.在播放器中需要植入广告动画。
- 11.需要有关于软件的介绍和简单使用方法。
- 12.用样式或主题控制字体与颜色。

14.1.1 需求解析：1.创建 Web 应用

- 创建项目 **Mp3Site**
- 在 **WebRoot** 或 **WebContent** 目录下面创建 **music** 子目录
- 在 **music** 目录中添加 **mp3** 文件与 **musics.xml**

参考项目： **MediaSite**

14.1.2 需求解析：2.编写 XML 文件

```
<?xml version="1.0" encoding="utf-8"?>
<musics>
    <music id="m_001">song.mp3</music>
    <music id="m_002">ILoveYou.mp3</music>
</musics>
```

参考项目： **MediaSite**

14.1.3 需求解析：3.网络下载 XML

```
URL url = new URL("http://192.168.50.50:8080/Mp3Site/music/musics.xml");
conn = url.openConnection();
is_sdcard = conn.getInputStream();
File destFile = new File("/sdcard/musics.xml");
destFile.createNewFile();
String path = destFile.getAbsolutePath();
FileOutputStream fos_sdcard = new FileOutputStream(destFile);
int bytesRead; byte buff[] = new byte[1024];
while ((bytesRead = is_sdcard.read(buff)) > 0) {
    fos_sdcard.write(buff, 0, bytesRead);
}
fos_sdcard.close(); is_sdcard.close();
```

参考： /SuperMediaPlayer/src/com/lxt008/common/DownloadFileActivity.java

14.1.4 需求解析：4.解析 XML 文件示

```
List<String> list = new ArrayList<String>();
/* 将 XML 转换成 Document 对象 */
DocumentBuilderFactory dbf =    DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
```

```
Document doc = db.parse(is);
NodeList nodeList = doc.getElementsByTagName(elementsTagName);
int len = nodeList.getLength();
for (int i = 0; i < len; i++) {
      String content = nodeList.item(i).getChildNodes().item(0)
      .getNodeValue();
      list.add(content);
}
```

参考：/SuperMediaPlayer/src/com/lxt008/common/XmlUtil.java

14.1.5 需求解析：5.下载 mp3

- 与下载 **XML** 文件类似。把 **mp3** 文件放到**/sdcard** 中。
- 把 **mp3** 文件名添加到 **ListView** 中。**Mp3** 文件多的话需要放到 **ScrollView** 中
- 参　　考　　：　　**/SuperMediaPlayer/src/com/lxt008/music/MusicActivity.java**（ **onListItemClick** 方法）与 **ScrollViewDemo**

14.1.6 需求解析：6.多线程下载 mp3

- 参　　考　　：　　**/SuperMediaPlayer/src/com/lxt008/music/MusicActivity.java**（ **onListItemClick** 方法）
- 在点击列表项的时候开一个线程去下载。
  - **onListItemClick** 方法中 **new Thread(Runnable).start()**

14.1.7 需求解析：7.显示 mp3 下载进度

- 在下载播放 **mp3** 的线程中通过 **Handler** 发下载状态的消息
- **handleMessage()**依据消息更新进度条。

参考：/SuperMediaPlayer/src/com/lxt008/music/MusicActivity.java

14.1.8 需求解析：8.在线播放 mp3

- 在线播放与下载后本地播放需要使用 **TabHost** 组织界面

```
mTabHost.addTab(mTabHost.newTabSpec("tab_test1")
        .setIndicator("TAB 1",getResources().getDrawable(R.drawable.img1))
         .setContent(new Intent(this,OtherActivity.class)));
```

- 参考项目：　**OnlineMp3** 与 **TabHostDemo**

14.1.9 需求解析：9.后台播放 mp3

- 需要在关闭播放器后音乐还能播放，所以要用 **Service** 来播放
- 参考：**/SuperMediaPlayer/src/com/lxt008/music/MusicService.java**

14.1.10 需求解析：10.植入广告

- 可以做成动画切换图片显示
- 参考项目： **AnimationDrawableDemo**

14.1.11 需求解析：11.关于对话框

- 加入“关于”、“帮助”等菜单。
- 点“关于”菜单中显示版权信息与软件简单介绍的对话框。
- 点“帮助”菜单中显示 **mp3** 播放器的简单用法的对话框。

  参考项目：**MenuDemo** 与 **DialogDemo**

14.1.12 需求解析：12.用主题控制字体与颜色

- **ThemeDemo\res\values** 下面放置 **themes.xml**
- 在 **themes.xml** 中定义应用全局的样式

  参考项目：**ThemeDemo**